Gould
Computer Systems

Diagnostic Manual

MP MASTER
MA34646

GOULD
Electronics

Gould CONCEPT/32™

Diagnostic Facility

Documentation Set

Book 3 of 5

Publication Order Number: 326-005450-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_06202

Gould CONCEPT/32™

Base Register

(CN.BRD.r)

Diagnostic Description

Diagnostic Number: 111031-231r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005530-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Base Register (CN.BRD.r) Diagnostic Description Release 1.0, Publication Order Number **326-005530-000,** was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-3/3-4
4-1/4-2
5-1/5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2
2.3 CPU Execution Modes 2-3
2.4 Memory Configuration 2-3

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2
3.3 Optional Programmed Halt 3-3
3.4 Unexpected Trap Messages 3-3

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test Structure 4-1

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 Base Register Diagnostic (CN.BRD.r) is a level one macrodiagnostic program that verifies the correct functioning of machine instruction in the base register mode.

The program consists of a sequence of tests which check the instructions in a logical manner, attempting not to use an instruction in a test sequence before verifying the instruction.

Options have been implemented that provide direct operator control over stop on error, loop on error, loop on test, and loop on subtest.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Base Register Diagnostic (CN.BRD.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires the following:

- CONCEPT/32 CPU
- IOP
- 128K words of memory

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 1 to 3 seconds to execute the first pass and 5 to 10 seconds for each subsequent 100 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Base Register Diagnostic (CN.BRD.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console and begins execution:

```
START CN.BRD.r (#111031-231r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to Chapter 5.

2.2 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Specify a pass count using the Diagnostic Overlay Loader Start command.
7-9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the operator's console upon completion of the first pass and every 100 passes thereafter.
11-12	Reset	Unimplemented.
13	Reset	Inhibits mapping through the CPIX. Perform mapping on every pass following pass one by resetting this option.
14	Reset	Inhibits the interval timer. Enable the interval timer on every pass, following pass two, by resetting this option.
15-18	Reset	Unimplemented.
19	Reset	Echoes all output to the line printer.
20-31	Reset	Unimplemented.

2.3 CPU Execution Modes

This program tests all CONCEPT/32 CPU execution modes. The following lists the execution modes under default conditions:

Pass #	Mapped Mode	Interval Timer
1	No	No
2	Yes	No
3-n	Yes	Yes

During execution, the program writes a message to the operator's console each time the execution mode changes.

Example

Setting option 14 at any point following the initial pass of the program causes the program to write the following message before each subsequent pass of the program:

```
MAPPED      INTERVAL TIMER OFF
```

2.4 Memory Configuration

The program dynamically determines memory size (starting at location zero) and writes the following message to the operator's console:

```
MEMORY CONFIGURATION:

(aaaaaa-aaaaaa) (mmmmKW)    PRESENT
(aaaaaa-aaaaaa) (mmmmKW) NON-PRESENT
```

aaaaaa — specifies an address range in memory that is either present or nonpresent, depending on the comment at the end of the message.

mmmm — specifies the size of the memory range in kilowords (KW).

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Base Register Diagnostic (CN.BRD.r), and the default option settings are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error message lists the failing test number, the pass count, and the program counter location where the error occurs. In addition, the error message lists information specific to the error condition. Section 3.2 contains the standard format for error messages generated during the execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at a location immediately following the location where the error occurs. If in subsequent passes the error continues to occur, refer to the program counter location in the source code listing to determine the cause of the error condition.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program write out one or more of the following messages:

```
TST:xxx, PASS:xxxxxx, PC:xxxxx
EXP  CC's:xxxx   ANS:xxxxxxxx
ACT  CC's:xxxx   ANS:xxxxxxxx
{MAPPED   }
{UNMAPPED } INTERVAL TIMER {OFF }
                           {ON  }
EXPECTED: xxxxxxxx, ACTUAL: xxxxxxxx, XOR: xxxxxxxx
VIOLATION INSTRUCTION ADDRESS: xxxxxxxx, ERROR COUNT: xxxxxxxx
VIOLATION INSTRUCTION WAS A STORE {FULLWORD }
                                  {HALFWORD }
CACHE MEMORY IS {ONLINE/ENABLED  }
                {OFFLINE/DISABLED}
```

TST	indicates the failing test number.
PASS	indicates the current program pass.
PC	indicates the program counter (PC) within the failing test.
EXP CC's	indicates software predicted (EXP) condition codes received from: • the expected result of an instruction under test • the expected result from a good comparison of data
ANS	indicates software predicted answer of instruction under test. It is stored in the location called EXPANS.
ACT CC's	indicates actual condition codes received from: • the result of executing an instruction under test • the result of the comparison of expected data and actual data
ANS	indicates the actual data from the instruction under test. It is stored in the location called ACTANS.
EXPECTED	indicates an expected hexadecimal value.
ACTUAL	indicates the actual hexadecimal value.
XOR	indicates the hexadecimal value returned when an exclusive OR is performed between the expected and actual values.
ERROR COUNT	indicates the number of errors detected.
{MAPPED / UNMAPPED}	indicates the program runs either mapped or unmapped.
INTERVAL TIMER {OFF / ON}	indicates the program runs with the interval either off or on.

3.3 Optional Programmed Halt

If option 2 (halt on error) is set, general purpose registers two through six are loaded wit execution status prior to the programmed halt as follows:

- GPR2 current pass count
- GPR3 current error count
- GPR4 60HZ clock count
- GPR5 arithmetic exception trap count
- GPR6 current test number in decimal

Option 2 does not affect GPRs 0, 1, or 7.

Entering the Panel mode Run command causes the program to reload the GPRs as they were at the time the error was detected, then halt program execution.

Entering the Panel mode Run command a second time causes the program to loop on error (if option 3 is set), or (if option 3 is reset) continue testing.

3.4 Unexpected Trap Messages

If option 2 is set, the program reports any unexpected traps detected by the program by issuing a programmed Halt and the following messages:

```
        UNEXPECTED  description  TRAP
OLD PSD:xxxxxxxxxxxxxxxx  TRAP ADDR:aaaa
TEST:tt,  PASS:cccccc
{MAPPED  }                {ON  }
{UNMAPPED} INTERVAL TIMER {OFF }
```

description — specifies the type of unexpected trap detected by the program.

xxx...xxx — specifies the value of the program status doubleword at the time the trap was detected.

aaaa — specifies the address of the unexpected trap.

tt — specifies the number of the test running when the trap was detected.

cccccc — specifies the current pass count.

{MAPPED / UNMAPPED} — indicates the mapping mode at the time the unexpected trap was detected.

INTERVAL TIMER {ON / OFF} — indicates the status of the interval timer (on or off) at the time the unexpected trap was detected.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould CONCEPT/32 Base Register Diagnostic (CN.BRD.r) contains 40 tests organized into four parts as follows:

- Tests 1 through 13 and 36 through 38 verify the ability of the base register logic to correctly execute all base mode only instructions.
- Tests 14 through 30 verify the ability of the base register logic to correctly execute all instructions which have a new opcode in base mode.
- Tests 31 and 32 verify the ability of the base register logic to identify opcodes which are undefined in base mode, but defined in non-base mode.
- Tests 33 through 35 verify the ability of the base register logic to correctly generate an address formed from the contents of the base register plus the contents of the index register plus the offset.
- Tests 39 and 40 verify the ability of the base register logic to properly handle pipeline conflict conditions.

4.2 Test Structure

Each test starts with a Branch and Link (BL) to a subroutine which controls the loop on test and subtest options. The subroutine also tests for proper sequencing of the test to ensure that the program counter (PC) is functioning properly, and not inadvertently skipping tests.

Each test consists of several subtests which check the instruction under test with various data patterns.

The basic composition of a subtest is as follows:

1. Set up the expected data and store it in EXPANS.
2. Execute an instruction.
3. Obtain the actual data and store it in ACTANS.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of Gould CONCEPT/32 Base Register Diagnostic (CN.BRD.r).

```
>LOAD CN.BRD.r
*LOADED: 'CN.BRD.r'
PGM: 03000
>STAR CN.BRD.r    PASS (100)

START CN.BRD.r (#111031-231r  date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

MEMORY CONFIGURATION:
(000000-1FFFFC)   (512KW)    PRESENT
(200000-FFFFFC)   (3584KW)   NON-PRESENT
UNMAPPED      INTERVAL TIMER OFF
END PASS:       1
MAPPED        INTERVAL TIMER OFF
MAPPED        INTERVAL TIMER ON

END CN.BRD.r:          100
```

Gould CONCEPT/32™

Cache/Shadow Memory

(CN.CMD.r)

Diagnostic Description

Diagnostic Number: 111031-252r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005590-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Cache/Shadow (CN.CMD.r) Diagnostic Description Release 1.0, Publication Order Number **326-005590-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1 through 1-2
2-1 through 2-6
3-1 through 3-3/3-4
4-1 through 4-4
5-1/5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-2

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Program Initialization 2-2
2.3 Option Set 2-5

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Header Information 3-2
3.3 Cache/Shadow Memory Contents 3-2
3.4 Test Result Reporting 3-2
3.5 Test 17 Error Message 3-3

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Summary of Execution Procedure 4-1
4.3 Test 1--Valid Configuration Test 4-2
4.4 Test 2--Data Storage RAM Check 4-2
4.5 Test 3--Data Storage RAMs Addressing Check 4-2
4.6 Test 4--Addressing Check 4-2
4.7 Test 5--Byte Addressing Check 4-2
4.8 Test 6--Halfword Addressing Check 4-2
4.9 Test 7--Doubleword Addressing Check 4-3
4.10 Test 8--Index Data Storage RAMs Addressing Check 4-3
4.11 Test 9--Validity RAMs Check 4-3
4.12 Test 10--Load File (LF) Speed Check 4-3
4.13 Test 11--Instruction Update Check 4-3
4.14 Test 12--Shadow Memory Initialization Check 4-3
4.15 Test 13--Shadow Memory/Cache Memory Interaction Check 4-4
4.16 Test 14--Cache I/O Miss Check 4-4
4.17 Test 15--Cache I/O Hit Check 4-4
4.18 Test 16--Shadow Memory I/O Read 4-4
4.19 Test 17--CONCEPT/32 Cache Instruction Verification 4-4

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 Cache/Shadow Memory Diagnostic (CN.CMD.r) is a level one macrodiagnostic program that contains progressively demanding cache/shadow memory tests. Tests 1 through 16 execute on the CONCEPT 32/87 and 32/97 processors; Test 17 executes on the CONCEPT 32/67 processor.

Options have been implemented to provide direct operator control over the tests to be executed, selection of the memory addressing modes, milestone messages, error messages, error looping, and other functions.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Cache/Shadow Memory Diagnostic (CN.CMD.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

The program requires a CONCEPT/32 CPU with a minimum of 64K words of memory. It runs in conjunction with Class F type I/O.

Note: When running this program on the CONCEPT 32/87, the firmware must be Version 9, Revision 1 or higher.

The program resides at a default starting address of 3000_{16}. The program must not be loaded into any memory module other than zero. The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

CONCEPT 32/67

This program requires approximately 1 second to execute the first pass and 3 to 5 seconds to execute each subsequent 10 passes depending on the following:

- machine configuration
- option setting

CONCEPT 32/87 and 32/97

This program requires approximately 1 to 2 minutes for each cache board bank and/or shadow board in the selected address mode, and an additional 1 to 3 minutes if I/O Tests are performed depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Cache/Shadow Memory Diagnostic (CN.CMD.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

Note: Instruction Stops (IS), Address Stops (AS), and Read Stops (RS) must not be used when executing this program as they disable cache and shadow memory.

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console:

```
START CN.CMD.r (#111031-252r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

When executed on a CONCEPT 32/87 or 32/97, with shadow memory, the program writes the following message to the operator's console for each shadow memory board present:

```
SHADOW MEMORY BOARD x ADDRESS: ********
```

x specifies the shadow memory board under test.

******** specifies the shadow memory board base address.

If no shadow memory is present, the program writes the following message to the operator's console:

```
NO SHADOW MEMORY DETECTED
```

2.2 Program Initialization

After issuing the start message, the program issues the following sequence of prompts to the operator's console:

1. The program accesses the CPU - Configuration Word to determine if shared memory exists. If there is no shared memory, the program begins the sequence with step 4. If the program determines that shared memory exists (bit 21 of CPU Configuration Word reset), the program issues the following prompt to the operator's console:

   SHARED MEMORY [Y, N] ?

   An invalid response (any character other than Y or N) causes the program to reissue the prompt.

   Responding with an N (no) causes the program to continue with step 4. Responding with a Y (yes) causes the program to issue the following prompt to the operator's console:

   ENTER LOWER ADDRESS

   Enter the hexadecimal value of the lowest address limit of shared memory. Entering a carriage return causes the program to reissue the prompt.

2. The program then issues the following prompt to the operator's console:

   ENTER HIGHER ADDRESS

   Enter the hexadecimal value of the highest address limit of shared memory. Entering a carriage return causes the program to reissue the prompt.

3. The program issues the following prompt to the operator's console:

   IS THE SYSTEM CONFIGURED WITH MULTIPROCESSOR SHARED MEMORY WITH CACHE WRITE THROUGH [Y, N] ?

   An invalid response (any character other than Y or N) causes the program to reissue the prompt.

   Responding with a Y (yes) causes the program to continue with step 4. Responding with an N (no) causes the program to examine the higher and lower addresses of shared memory to determine if the shared memory is in shadow memory range. If so, the program writes the following message to the operator's console:

   INVALID CONFIGURATION! THE SYSTEM IS NOT CONFIGURED WITH MULTIPROCESSOR SHARED MEMORY WITH CACHE WRITE THROUGH. THE SHARED MEMORY IS LOCATED IN THE SHADOW MEMORY ADDRESS RANGE WHICH IS INVALID.

   At this point the program halts.

4. The program issues the following prompt to the operator's console:

   IS THE SYSTEM CONFIGURED WITH A VECTOR PROCESSOR [Y, N] ?

   An invalid response (any character other than Y or N) causes the program to reissue the prompt.

   Responding with an N (no) causes the program to continue with step 5 and prompt for I/O tests. Responding with a Y (yes) causes the program to prompt for the vector processor Low and High memory addresses and write the two following prompts, each requiring their own response, to the operator's console:

   ENTER THE VECTOR PROCESSOR LOW MEMORY ADDRESS
   ENTER THE VECTOR PROCESSOR HIGH MEMORY ADDRESS

   The program examines the lower and higher vector processor memory addresses and determines if the vector processor memory resides in shadow memory. If so, the program writes the following message to the operator's console:

   INVALID CONFIGURATION! THE VECTOR PROCESSOR MEMORY ADDRESSES ARE LOCATED IN A SHADOW MEMORY RANGE.

   At this point the program halts.

5. The program issues the following prompt to the operator's console:

   EXECUTE I/O TESTS [Y, N] ?

   An invalid response (any character other than Y or N) causes the program to reissue the prompt.

   Responding with a Y (yes) causes the program to prepare to execute its I/O tests (Tests 14, 15, and 16). Responding with an N (no) causes the program to bypass I/O testing, which includes bypassing steps 6 and 7 of the initialization sequence.

   Note: I/O Tests cannot be executed from a command file.

6. If a Y is entered in response to the I/O test prompt, the program issues the following prompt to the operator's console:

   ENTER CHANNEL-SUBADDRESS OF FLOPPY DISC [7EF0, 7EF1, XXFX] ?

   Enter the device subaddress corresponding to the floppy disc drive to be tested. On a system with only one floppy disc controller, the valid device subaddresses are either F0 or F1.

7. The program issues the following prompt to the operator's conosle:

   INSERT FORMATTED SCRATCH DISKETTE IN nnnn READY [Y, N] ?

   nnnn specifies the value assigned in response to the channel-subaddress prompt.

   Insert a formatted, write-enabled diskette into the drive specified in the prompt. When ready, enter a Y (yes) on the operator's console to complete the initialization sequence and begin the program execution. Responding with an N (no) causes the program to reissue the prompt until a Y is entered.

2.3 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader Start command.
7-9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the line printer upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Executes tests in unmapped mode only.
13	Reset	Executes tests in unmapped, extended addressing mode only.
14	Reset	Executes tests in mapped mode only.
15	Reset	Executes tests in mapped, extended addressing mode only.

Option	Default	Effect When Set
16	Reset	Inhibits execution of I/O tests (Tests 14, 15, and 16). This option has no effect when executing the program from a command file. The program writes the following message to the operator's console when executing from the Diagnostic Overlay Loader: EXECUTE I/O TESTS [N,Y]?
17	Reset	Unimplemented.
18	Reset	Inhibits error message expansion.
19	Reset	Echoes all messages to the line printer.
20	Reset	Inhibits testing buffer board 0, bank 0.
21	Reset	Inhibits testing buffer board 1, bank 0.
22	Reset	Inhibits testing buffer board 2, bank 0.
23	Reset	Inhibits testing buffer board 3, bank 0.
24	Reset	Inhibits testing buffer board 0, bank 1.
25	Reset	Inhibits testing buffer board 1, bank 1.
26	Reset	Inhibits testing buffer board 2, bank 1.
27	Reset	Inhibits testing buffer board 3, bank 1.
28	Reset	Writes scratchpad image.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Cache/Shadow Memory Diagnostic (CN.CMD.r), and the default options are set, the program writes out an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

Error messages written to the operator's console for the first 16 tests are organized into three areas:

- error message header
- cache/shadow memory contents
- test results

Refer to Sections 3.2 through 3.4 for a description of these areas.

The error message written for Test 17 contains the test number, pass count, program counter, and expected and actual data. Refer to Section 3.5 for a description of the error message generated for Test 17.

3.2 Header Information

The first four lines of the error message contain the following information:

```
TEST: xx  PSD: xxxxxxxx-xxxxxxxx
END PASS: xxxxx
CPU MODE   xxxxxxxx
message
```

TEST	indicates the number of the failing test.
PSD	indicates the program status doubleword immediately after the failing instruction.
END PASS	indicates the current program pass.
CPU MODE	indicates the mode of execution (i.e., mapped or unmapped).
message	specifies a brief message about the error.

3.3 Cache/Shadow Memory Contents

The next lines of the error message contain a list of addresses corresponding to cache/shadow memory contents. These addresses are identified by a title and their bit positions.

```
TEST ADDRESS:                xxxxxx     BITS  8-31:xxxxxxxxxxxxxxxxxxxxxxxx
COLUMN ADDRESS:                 xxx     BITS  8-18:xxxxxxxxxxx
ROW ADDRESS:                     xx     BITS 19-26:          xxxxxxxx
QUARTER BLOCK ADDRESS:            x     BITS 27-28:                  xx
DATA STORAGE ADDRESS:           xxx     BITS 19-28:          xxxxxxxxxx
EVENWORD                                BIT     29:                    x
C BITS          x                       BITS 30-31:                     xx
F BIT           x
```

TEST ADDRESS	indicates the address being tested at the time the error occurred. This address is specified in both hexadecimal and binary.
COLUMN ADDRESS	indicates a breakdown of the test address into the corresponding locations within cache memory.

3.4 Test Result Reporting

The last two lines of the message report the expected and actual result of executing the failing instruction.

```
EXP DATA: xxxxxxxx
REC DATA: xxxxxxxx
```

EXP DATA:	indicates the expected data.
REC DATA:	indicates the actual data received.

3.5 Test 17--Error Message

The error message for Test 17 follows:

```
TST:17, PASS:xxxxxx, PC:xxxxx
EXP CC's:xxxx  ANS:xxxxxxxx
ACT CC's:xxxx  ANS:xxxxxxxx
CACHE MEMORY CONTROL GPR CONTENTS:xxxxxxxx
...HALTED ON ERROR--'RUN' FOR GPR's
```

PASS	indicates the current program pass.
PC	indicates the program counter (PC) within the failing test.
EXP CC's	indicates software predicted (EXP) condition codes received from the following: . the expected result of an instruction under test. . the expected result from a good comparison of data.
ANS	indicates software predicted answer of instruction under test. It is stored in the location called EXPANS.
ACT CC's	indicates actual received condition codes from the following: . the result of executing an instruction under test. . the result of the comparison of expected data and actual data.
ANS	indicates the actual data from the instruction under test. It is stored in the location called ACTANS.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

The Gould Cache/Shadow Memory Diagnostic (CN.CMD.r) level one macrodiagnostic program contains 17 tests which check the cache/shadow memory for CONCEPT 32/67, 32/87, and 32/97 processors. Tests 1 through 16 execute on the CONCEPT 32/87 and 32/97; Test 17 executes on the CONCEPT/32/67.

This program tests each cache buffer board and each shadow memory board separately. Cache buffer slot (B0) must be filled with a cache buffer board. A combination of up to two shadow memory boards can be inserted into either cache buffer slots (B1), (B2), or (B3) per processor.

4.2 Summary of Execution Procedure

The program execution procedure verifies that data is being written to and read from cache/shadow memory boards correctly when the board under test is enabled.

The program disables all boards except the one under test then determines the board type. The program also determines the base address if the board under test is a shadow memory board.

The program writes a data pattern into both the board under test and main memory then disables the board. The program then writes a one's complemented data pattern to the disabled board and main memory.

When the enabled board under test is read, the data should be the original data pattern. If the one's complemented data pattern is received, the data came from main memory.

4.3 Test 1--Valid Configuration Test

This test verifies that the cache and/or shadow memory boards are configured properly.

Note: If one of the shadow memory boards has a base address at zero, the test notifies the operator that only the unused portion of memory located in that first block of shadow memory is tested.

This test is invalid for the CONCEPT 32/67 processor.

4.4 Test 2--Data Storage RAM Check

This test verifies that the cache/shadow memory data storage RAMs can be properly loaded and read.

This test is invalid for the CONCEPT 32/67 processor.

4.5 Test 3--Data Storage RAMs Addressing Check

This test verifies the cache memory and shadow memory data storage RAMs addressing capabilities on a single cache buffer board or shadow memory board.

This test is invalid for the CONCEPT 32/67 processor.

4.6 Test 4--Addressing Check

This test verifies correct data writes and reads using 2K increments.

This test is not valid for shadow memory. This test is invalid for the CONCEPT 32/67 processor.

4.7 Test 5--Byte Addressing Check

This test verifies correct data writes and reads using byte increments. When testing a cache buffer board, the first available 2K of memory following the program is loaded with a binary progression data pattern in byte increments.

This test is invalid for the CONCEPT 32/67 processor.

4.8 Test 6--Halfword Addressing Check

This test verifies correct data writes and reads using halfword increments. When testing a cache buffer board, the test loads the first available 2K of memory following the program with a binary progression data pattern in halfword increments.

This test is invalid for the CONCEPT 32/67 processor.

4.9 Test 7—Doubleword Addressing Check

This verifies correct data writes and reads using doubleword increments. The test uses a binary progression data pattern in doubleword increments.

This test is invalid for the CONCEPT 32/67 processor.

4.10 Test 8—Index Data Storage RAMs Addressing Check

This test verifies the cache memory index RAMs addressing capabilities. Each location of the index RAM is tested individually with a fixed data pattern.

This test is not valid for shadow memory. This test is invalid for the CONCEPT 32/67 processor.

4.11 Test 9—Validity RAMs Check

This test verifies the cache memory validity RAMs addressing capabilities.

This test is invalid for shadow memory. This test is invalid for the CONCEPT 32/67 processor.

4.12 Test 10—Load File (LF) Speed Check

This test verifies cache memory and shadow memory timing at high speed execution using the Load File (LF) instruction.

This test is invalid for the CONCEPT 32/67 processor.

4.13 Test 11—Instruction Update Check

This test verifies that when cache memory is operating in a normal mode (not instruction bypass mode), the instructions are updated in the cache memory data storage RAMs (one buffer board tested at a time) as they are executed.

This test is not valid for shadow memory boards. This test is invalid for the CONCEPT 32/67 processor.

4.14 Test 12—Shadow Memory Initialization Check

This test verifies the successful operation of the initialization bits for the shadow memory board under test.

This test is invalid for the CONCEPT 32/67 processor.

4.15 Test 13--Shadow Memory/Cache Memory Interaction Check

This test checks for shadow memory and cache memory interaction using an address in address data pattern with all cache and shadow memory boards on-line.

This test is invalid for the CONCEPT 32/67 processor.

4.16 Test 14--Cache I/O Miss Check

This test verifies that a miss during a Read or Write operation from main memory to I/O does not update or invalidate cache memory (one buffer board tested at a time).

This test is invalid for shadow memory boards. This test is invalid for the CONCEPT 32/67 processor.

4.17 Test 15--Cache I/O Hit Check

This test verifies that a hit during a Write operation from main memory to I/O does not update or invalidate cache memory, but a Read operation from I/O to memory does invalidate cache (one buffer board tested at a time).

This test is invalid for shadow memory boards. This test is invalid for the CONCEPT 32/67 processor.

4.18 Test 16--Shadow Memory I/O Read

This test verifies that a Write operation from shadow memory to I/O does not invalidate shadow memory and that shadow memory is updated with main memory as a result of a Read operation from I/O with the shadow board enabled.

This test is invalid for the CONCEPT 32/67 processor.

4.19 Test 17--CONCEPT 32/67 Cache Instruction Verification

This test verifies that the operand cache and the instruction cache are functioning properly on a CONCEPT 32/67 processor.

This test is invalid for the CONCEPT 32/87 and 32/97 processors.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of the Gould Cache/Shadow Memory Diagnostic (CN.CMD.r).

```
>LOAD CN.CMD.r
*LOADED: 'CN.CMD.r'
PGM: 03000
>STAR CN.CMD.r  PASS (1)


START CN.CMD.r (#111031-252r    date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

CPU MODE:    UNMAPPPED
(TEST 1-16 BYPASSED - N/A THIS MACHINE TYPE)
TEST 17 - 32/67 CACHE INSTRUCTION VERIFICATION

CPU MODE:    EXTENDED
(TEST 1-16 BYPASSED - N/A THIS MACHINE TYPE)
TEST 17 - 32/67 CACHE INSTRUCTION VERIFICATION

CPU MODE:    MAPPED
(TEST 1-16 BYPASSED - N/A TIS MACHINE TYPED)
TEST 17 - 32/67 CACHE INSTRUCTION VERIFICATION

CPU MODE:    MAPPED EXTENDED
(TEST 1-16 BYPASSED - N/A THIS MACHINE TYPE)
TEST 17 - 32/67 CACHE INSTRUCTION VERIFICATION

END PASS:            1

END CN.CMD.r                    1 PASS(ES)
```

Gould CONCEPT/32™

Central Processing Unit - Part One

(CN.CP1.r)

Diagnostic Description

Diagnostic Number: 111031-210r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005500-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Central Processing Unit Diagnostic - Part One (CN.CP1.r) Diagnostic Description Release 1.0, Publication Order Number **326-005500-000,** was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-2
3-1 through 3-3/3-4
4-1 through 4-4
5-1/5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2
3.3 Optional Programmed Halt 3-3

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test Structure 4-4

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 CPU Diagnostic - Part One (CN.CP1.r) is a level one macrodiagnostic program that verifies the correct functioning of CONCEPT/32 central processing unit (CPU) machine instructions.

The program consists of a sequence of tests which check the instructions in a logical manner, attempting not to use an instruction in a test sequence before verifying the instruction. An extensive condition code bit analysis is done on both the immediate result of the instruction under test and the result of the comparison of actual and expected data.

Options have been implemented that provide direct operator control over stop on error, loop on error, loop on test, and loop on subtest.

Note: This manual describes the specific attributes of Gould CONCEPT/32 CPU Diagnostic - Part One (CN.CP1.r). For a discussion of level one macrodiagnostic program attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU, an IOP, and a minimum of 128K words of memory.

The program resides at a default starting address of 3000_{16}. The program must not be loaded into any memory module other than zero.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 1 to 10 seconds to execute each 100 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 CPU Diagnostic - Part One (CN.CP1.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console and begins execution:

```
START CN.CP1.r (#111031-210r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE=32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to Chapter 5.

2.2 Option Set

The following options are implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Specify a pass count using the Diagnostic Overlay Loader Start command.
7-9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the operator's console upon completion of the first pass and every 100 passes thereafter.
11-18	Reset	Unimplemented.
19	Reset	Echoes all output to the line printer.
20-25	Reset	Unimplemented.
26	Reset	Causes the program to immediately halt upon error detection to allow for a snapshot of the error location. The program does not write an error message or save the contents of the GPRs. Nullify the halt by entering the Panel mode Run command on the operator's console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27-31	Reset	Reserved - Do not use.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 CPU Diagnostic - Part One (CN.CP1.r), and the default option settings are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error message lists the failing test number, the pass count, and the program counter location where the error occurred. In addition, the error message lists information specific to the error condition. Section 3.2 contains the standard format for error messages generated during the execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at a location immediately following the location where the error occurred. If in subsequent passes the error continues to occur, refer to the program counter location in the source code listing to determine the cause of the error condition.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program write out one or more of the following messages:

```
TST:xxx, PASS:xxxxxxx, PC:xxxxx
EXP  CC's:xxxx   ANS:xxxxxxxx xxxxxxxx
ACT  CC's:xxxx   ANS:xxxxxxxx xxxxxxxx
EXPECTED: xxxxxxxx,  ACTUAL: xxxxxxxx,  XOR: xxxxxxxx
VIOLATION INSTRUCTION ADDRESS: xxxxxxxx,  ERROR COUNT: xxxxxxxx
VIOLATION INSTRUCTION WAS A STORE  {FULLWORD}
                                   {HALFWORD}
CACHE MEMORY IS  {ONLINE/ENABLED  }
                 {OFFLINE/DISABLED}
```

TST	indicates the failing test number.
PASS	indicates the current program pass.
PC	indicates the program counter (PC) within the failing subtest.
EXP CC's	indicates software predicted (EXP) condition codes received from: . the expected result of an instruction under test . the expected result from a good comparison of data (usually CC=0001)
ANS	indicates the software predicted answer resulting from the instruction under test. Given in doubleword format, using GPR2 and GPR3, for simplicity of error reporting.
ACT CC's	indicates the actual received condition codes from: . the result of executing the instruction under test . the result of the comparison of expected data and actual data
ANS	indicates the actual resulting data from the instruction under test, as recorded in GPR6 and GPR7.
EXPECTED	indicates an expected hexadecimal value.
ACTUAL	indicates the actual hexadecimal value.
XOR	indicates the hexadecimal value returned when an exclusive or is performed between the expected and actual values.
ERROR COUNT	indicates the number of errors detected.

3.3 Optional Programmed Halt

If option 2 (halt on error) is set, the general purpose registers are loaded with execution status prior to a programmed Halt as follows:

- GPR2 current pass count
- GPR3 current error count
- GPR4 60HZ clock count
- GPR5 arithmetic exception trap count
- GPR6 current test number in decimal

Option 2 does not affect GPRs 0, 1, and 7.

Entering the Panel mode Run command causes the program to reload the GPRs as they were at the time the error was detected, then halt program execution.

Entering the Panel mode Run command a second time causes the program to loop on error (if option 3 is set), or (if option 3 is reset) continue testing.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould CONCEPT/32 CPU Diagnostic - Part One (CN.CP1.r) contains two parts: an initialization procedure performed by the program and the instruction tests.

The initialization procedure checks the instructions which are commonly used throughout the program, condition code setting and branching, and the general purpose registers (GPRs). Errors generated during the initialization procedure immediately halt the processor.

There are 130 instruction tests that verify CONCEPT/32 machine instructions. The tests are executed in the following sequence:

TEST 1: LOAD BYTE (LB)
TEST 2: LOAD HALFWORD (LH)
TEST 3: LOAD DOUBLEWORD (LD)
TEST 4: LOAD MASKED WORD (LMW)
TEST 5: LOAD MASKED DOUBLEWORD (LMD)
TEST 6: LOAD MASKED HALFWORD (LMH)
TEST 7: LOAD MASKED BYTE (LMB)
TEST 8: LOAD IMMEDIATE (LI)
TEST 9: LOAD NEGATIVE WORD (LNW)
TEST 10: LOAD NEGATIVE BYTE (LNB)
TEST 11: LOAD NEGATIVE HALFWORD (LNH)
TEST 12: LOAD NEGATIVE DOUBLEWORD (LND)
TEST 13: STORE BYTE (STB)
TEST 14: STORE HALFWORD (STH)
TEST 15: STORE DOUBLEWORD (STD)
TEST 16: STORE MASKED WORD (STMW)
TEST 17: STORE MASKED BYTE (STMB)
TEST 18: STORE MASKED HALFWORD (STMH)
TEST 19: STORE MASKED DOUBLEWORD (STMD)
TEST 20: TEST BIT IN REGISTER (TBR)
TEST 21: TEST BIT IN MEMORY (TBM)
TEST 22: SET BIT IN REGISTER (SBR)
TEST 23: ZERO BIT IN REGISTER (ZBR)
TEST 24: SET BIT IN MEMORY (SBM)
TEST 25: ZERO BIT IN MEMORY (ZBM)
TEST 26: ADD BIT IN REGISTER (ABR)
TEST 27: ADD BIT IN MEMORY (ABM)
TEST 28: BRANCH CONDITION FALSE (BCF)
TEST 29: BRANCH CONDITION TRUE (BCT)
TEST 30: BRANCH AFTER INCREMENTING BYTE (BIB)

TEST 31: BRANCH AFTER INCREMENTING HALFWORD (BIH)
TEST 32: BRANCH AFTER INCREMENTING WORD (BIW)
TEST 33: BRANCH AFTER INCREMENTING DOUBLEWORD (BID)
TEST 34: ZERO MEMORY BYTE (ZMB)
TEST 35: ZERO MEMORY HALFWORD (ZMH)
TEST 36: ZERO MEMORY WORD (ZMW)
TEST 37: ZERO MEMORY DOUBLEWORD (ZMD)
TEST 38: ZERO REGISTER (ZR)
TEST 39: COMPARE ARITH WITH MEM BYTE (CAMB)
TEST 40: COMPARE ARITH WITH MEM HALFWORD (CAMH)
TEST 41: COMP ARITH WITH MEM WORD (CAMW)
TEST 42: COMP ARITH WITH MEM DOUBLEWORD (CAMD)
TEST 43: COMPARE IMMEDIATE (CI)
TEST 44: COMPARE MASKED WITH MEM WORD (CMMW)
TEST 45: COMPARE MASKED WITH MEM BYTE (CMMB)
TEST 46: COMP MASKED WITH MEM HALFWORD (CAMH)
TEST 47: COMP MASKED WITH MEM DOUBLEWORD (CAMD)
TEST 48: (CAR) & (CMR)
TEST 49: AND MEMORY WORD (ANMW)
TEST 50: AND MEMORY BYTE (ANMB)
TEST 51: AND MEMORY HALFWORD (ANMH)
TEST 52: AND MEMORY DOUBLEWORD (ANMD)
TEST 53: AND REGISTER AND REGISTER (ANR)
TEST 54: OR MEMORY WORD (ORMW)
TEST 55: OR MEMORY BYTE (ORMB)
TEST 56: OR MEMORY HALFWORD (ORMH)
TEST 57: OR MEMORY DOUBLEWORD (ORMD)
TEST 58: OR REGISTER AND REGISTER (ORR)
TEST 59: OR REGISTER AND REGISTER MASKED (ORRM)
TEST 60: EXCLUSIVE OR MEMORY WORD (EOMW)
TEST 61: EXCLUSIVE OR MEMORY BYTE (EOMB)
TEST 62: EXCLUSIVE OR MEM HALFWORD (EOMH)
TEST 63: EXLCUSIVE OR MEM DOUBLEWORD (EOMD)
TEST 64: EXCLUSIVE OR REG AND REG (EOR)
TEST 65: EXCLUSIVE OR REG AND REG MASKED (EORM)
TEST 66: LOAD ADDRESS (LA, LEA, LEAR)
TEST 67: TRANSFER REGISTER TO REGISTER (TRR)
TEST 68: TRANSFER REG TO REG MASKED (TRRM)
TEST 69: TRANSFER REGISTER NEGATIVE (TRN)
TEST 70: TRANSFER REG NEGATIVE MASKED (TRNM)
TEST 71: TRANSFER REGISTER COMPLEMENT (TRC)
TEST 72: TRANSFER REG COMP MASKED (TRMC)
TEST 73: EXCHANGE REGISTER (XCR)
TEST 74: EXCHANGE REGISTER MASKED (XCRM)
TEST 75: TRANSFER REGISTER TO PSWR (TRSW)
TEST 76: SHIFT LEFT LOGICAL (SLL)
TEST 77: SHIFT RIGHT LOGICAL (SRL)
TEST 78: SHIFT LEFT CIRCULAR (SLC)
TEST 79: SHIFT RIGHT CIRCULAR (SRC)
TEST 80: SHIFT LEFT ARITHMETIC (SLA)
TEST 81: SHIFT RIGHT ARITHMETIC (SRA)
TEST 82: SHIFT LEFT LOGICAL DOUBLE (SLLD)
TEST 83: SHIFT RIGHT LOGICAL DOUBLE (SRLD)

TEST 84: SHIFT LEFT ARITH DOUBLE (SLAD)
TEST 85: SHIFT RIGHT ARITH DOUBLE (SRAD)
TEST 86: SHIFT AND COUNT ZEROS (SCZ)
TEST 87: NORMALIZE (NOR)
TEST 88: NORMALIZE DOUBLE (NORD)
TEST 89: STORE FILE (STF) / LOAD FILE (LF)
TEST 90: ADD REGISTER TO REGISTER (ADR)
TEST 91: ADD REG TO REG MASKED (ADRM)
TEST 92: ADD MEMORY WORD (ADMW)
TEST 93: ADD MEMORY BYTE (ADMB)
TEST 94: ADD MEMORY HALFWORD (ADMH)
TEST 95: ADD MEMORY DOUBLEWORD (ADMD)
TEST 96: ADD REGISTER TO MEMORY WORD (ARMW)
TEST 97: ADD REGISTER TO MEMORY BYTE (ARMB)
TEST 98: ADD REG TO MEMORY HALFWORD (ARMH)
TEST 99: ADD REG TO MEMORY DOUBLEWORD (ARMD)
TEST 100: ADD IMMEDIATE (ADI)
TEST 101: SUBTRACT REGISTER FROM REGISTER (SUR)
TEST 102: SUBTRACT REG FROM REG MASKED (SURM)
TEST 103: SUBTRACT MEMORY WORD (SUMW)
TEST 104: SUBTRACT MEMORY BYTE (SUMB)
TEST 105: SUBTRACT MEMORY HALFWORD (SUMH)
TEST 106: SUBTRACT MEMORY DOUBLEWORD (SUMD)
TEST 107: SUBTRACT IMMEDIATE (SUI)
TEST 108: MULTIPLY BY MEMORY WORD (MPMW)
TEST 109: MULTIPLY BY MEMORY BYTE (MPMB)
TEST 110: MULTIPLY BY MEMORY HALFWORD (MPMH)
TEST 111: MULTIPLY REG BY REG (MPR)
TEST 112: MULITPLY IMMEDIATE (MPI)
TEST 113: DIVIDE BY MEMORY WORD (DVMW)
TEST 114: DIVIDE BY MEMORY BYTE (DVMB)
TEST 115: DIVIDE BY MEMORY HALFWORD (DVMH)
TEST 116: DIVIDE REGISTER BY REGISTER (DVR)
TEST 117: DIVIDE IMMEDIATE (DVI)
TEST 118: EXTEND SIGN (ES)
TEST 119: ROUND REGISTER (RND)
TEST 120: BRANCH FUNCTION TRUE (BFT)
TEST 121: TRANSFER SCRATCHPAD TO REG (TRSC)
TEST 122: LOAD CONTROL SWITCHES (LCS)
TEST 123: EXECUTE HALFWORD INSTRUCTIONS BACK-TO-BACK
TEST 124: EXECUTE REGISTER (EXR)
TEST 125: EXECUTE REGISTER RIGHT (EXRR)
TEST 126: EXECUTE MEMORY (EXM)
TEST 127: INDIRECT ADDRESSING TEST
TEST 128: PIPELINE REGISTER CONFLICTS
TEST 129: PIPELINE CONFLICT VIOLATIONS
TEST 130: INTEGER MULTIPLICATION SIMULATION

This program does not test Trap, Interrupt, I/O, Map, or Floating Point Arithmetic instructions.

4.2 Test Structure

Each test starts with a Branch and Link (BL) to a subroutine which controls the loop on test and subtest options. The subroutine also tests for proper sequencing of the test to ensure that the program counter (PC) is functioning properly, and not inadvertently skipping tests.

Each test consists of several subtests which check the instruction under test with various data patterns and ensure the accuracy of the condition codes.

The basic composition of a subtest is as follows:

1. Load GPR2 and GPR3, if needed, with the expected results.
2. Set up any operands, usually in GPR6 and GPR7, with test data.
3. Execute the instruction under test.

Following execution of the first pass, the program enables the system integrity traps and the Real-Time Clock interrupt (RPL18).

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of Gould CONCEPT/32 CPU Diagnostic - Part One (CN.CP1.r).

```
>LOAD CN.CP1.r
*LOADED: 'CN.CP1.r'
 PGM: 03000
>STAR CN.CP1.r PASS(100)

START CN.CP1.r (#111031-210r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

END PASS:      1
END PASS:    100

END CN.CP1.r:  100 PASS(ES)
```

Gould CONCEPT/32™

Central Processing Unit - Part Two

(CN.CP2.r)

Diagnostic Description

Diagnostic Number: 111031-211r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005510-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Central Processing Unit Diagnostic - Part Two (CN.CP2.r) Diagnostic Description Release 1.0, Publication Order Number **326-005510-000,** was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
History page
iii/iv through v/vi
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-2
4-1 through 4-2
5-1/5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2
3.3 Optional Programmed Halt 3-2

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test Structure 4-2

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

The Gould CONCEPT/32 CPU Diagnostic - Part Two (CN.CP2.r) is a level one macrodiagnostic program that verifies the correct functioning of CONCEPT/32 central processing unit (CPU) machine instructions.

The program consists of a sequence of tests which check the instructions in a logical manner, attempting not to use an instruction in a test sequence before verifying the instruction. An extensive condition code bit analysis is done on both the immediate result of the instruction under test and the result of the comparison of actual and expected data.

Options have been implemented that provide direct operator control over stop on error, loop on error, and loop on subtest.

Note: This manual describes the specific attributes of Gould CONCEPT/32 CPU Diagnostic - Part Two (CN.CP2.r). For a discussion of level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU, an IOP, and a minimum of 64K words of memory.

The program resides at a default starting address of 3000_{16}. The program must not be loaded into any memory module other than zero.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 1 to 10 seconds to execute each 100 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 CPU Diagnostic - Part Two (CN.CP2.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console and begins execution:

```
START CN.CP2.r (#111031-211r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to Chapter 5.

2.2 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Unimplemented.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Specify a pass count using the Diagnostic Overlay Loader START command.
7-9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the operator's console upon completion of the first pass and every 100 passes thereafter.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Inhibits Extended Addressing. Perform extended addressing on every pass following pass one by resetting this option.
13	Reset	Inhibits Mapped mode. Set mapped mode for every pass following pass two by resetting this option.
14	Reset	Inhibits the interval timer. Activate the interval timer for every pass following pass three by resetting this option.

Option	Default	Effect When Set
15-18	Reset	Unimplemented.
19	Reset	Echoes all output to the line printer.
20	Reset	Unimplemented.
21	Reset	Writes a scratchpad image.
22-25	Reset	Unimplemented.
26	Reset	Causes the program to immediately halt upon error detection to allow for a snapshot of the error location. The program does not write an error message or save the contents of the GPRs. Nullify the halt by entering the Panel mode Run command on the operator's console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27-31	Reset	Reserved - Do not use.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 CPU Diagnostic - Part Two (CN.CP2.r), and the default option settings are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error message lists the pass count and the program counter location where the error occurred. In addition, the error message lists information specific to the error condition. Section 3.2 contains the standard format for error messages generated during the execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at a location immediately following the location where the error occurred. If in subsequent passes the error continues to occur, refer to the program counter location in the source code listing to determine the cause of the error condition.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program write out one or more of the following messages:

```
ERROR/PASS:xxxxxxxx, PC:xxxxx
EXP  CC's:xxxx   ANS:xxxxxxxx xxxxxxxx
ACT  CC's:xxxx   ANS:xxxxxxxx xxxxxxxx
```

ERROR/PASS — indicates the current program pass.

PC — indicates the program counter (PC) within the failing subtest.

EXP CC's — indicates software predicted (EXP) condition codes received from:

- the expected result of an instruction under test.
- the expected result from a good comparison of data (usually CC=0001).

ANS — indicates the software predicted answer resulting from the instruction under test. Given in doubleword format, using GPR2 and GPR3, for simplicity of error reporting.

ACT CC's — indicates the actual received condition codes from:

- the result of executing the instruction under test.
- the result of the comparison of expected data and actual data.

ANS — indicates the actual resulting data from the instruction under test, as recorded in GPR6 and GPR7.

3.3 Optional Programmed Halt

If option 2 (halt on error) is set, the general purpose registers are loaded with execution status prior to the programmed halt as follows:

- GPR2 current pass count
- GPR3 current error count
- GPR4 60HZ clock count
- GPR5 arithmetic exception trap count
- GPR6 current test number in decimal

Option 2 does not affect GPRs 0, 1, and 7.

Entering the Panel mode Run command causes the program to reload the GPRs as they were at the time the error was detected, then halt program execution.

Entering the Panel mode Run command a second time causes the program to loop on error (if option 3 is set), or (if option 3 is reset) continue testing.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould CONCEPT/32 CPU Diagnostic - Part Two (CN.CP2.r) contains 94 tests which verify the machine instruction set. The program tests the instructions in the following sequence:

- Test Bit In Register
- Exclusive Or Memory Word
- Add Bit In Register
- Test Bit In Memory
- Load Word
- Register Selection Tests
- Load Immediate
- Load Doubleword
- Load Halfword
- Load Byte
- Load Masked Word
- Load Masked Doubleword
- Load Masked Halfword
- Load Masked Byte
- Exclusive Or Memory Doubleword
- Exclusive Or Memory Halfword
- Exclusive Or Memory Byte
- Transfer Registers To Register
- Exchange Registers
- Transfer Register Complement
- Transfer Register To Status Word
- Transfer Register To Register Masked
- Transfer Register Complement Masked
- Exchange Register Masked
- Transfer Register Negative
- Transfer Register Negative Masked
- Store Byte
- Store Halfword
- Store Doubleword
- Store Masked Word
- Store Masked Byte
- Store Masked Halfword
- Store Masked Doubleword
- Zero Register
- Zero Memory Word
- Zero Memory Doubleword
- Zero Memory Byte
- Zero Memory Halfword
- Exclusive Or Register And Register
- Exclusive Or Register And Register Masked
- Compare Masked With Memory Word
- Compare Masked With Memory Halfword
- Compare Masked With Memory Byte
- Compare Masked With Memory Doubleword
- Compare Masked Register
- And Memory Word
- And Memory Doubleword
- And Memory Halfword
- And Memory Byte
- And Register And Register
- Or Memory Word
- Or Register And Register
- Or Register And Register Masked
- Or Memory Halfword
- Or Memory Byte
- Or Memory Doubleword
- Branch Condition True
- Branch Condition False

- Branch Function True
- Zero Bit In Register
- Set Bit In Register
- Zero Bit In Memory
- Set Bit In Memory
- Add Bit In Memory
- Shift And Circular
- Shift Right Circular
- Shift Right Logical
- Shift Left Logical
- Shift Right Arithmetic
- Shift Left Arithmetic
- Shift Left Logical Double
- Shift Right Arithmetic Double
- Shift Left Arithmetic Double
- Shift Right Logical Double
- Shift And Count Zeros
- Round Register
- Normalize
- Normalize Double
- Extended Sign
- Load Effective Address
- Branch And Link
- Load File
- Store File
- Execute Register
- Execute Register Right
- Execute Memory
- Branch After Incrementing Byte
- Branch After Incrementing Halfword
- Branch After Incrementing Word
- Branch After Incrementing Doubleword
- Load Negative Word
- Load Negative Byte
- Load Negative Halfword
- Load Negative Doubleword

The program does not test Trap, Interrupt, I/O, Map, Fixed Point Arithmetic, Indexing, Indirect Addressing or Floating Point Arithmetic instructions.

4.2 Test Structure

Each test starts with a Branch and Link (BL) to a subroutine which controls the loop on test and subtest options. The subroutine also tests for proper sequencing of the test to ensure that the program counter (PC) is functioning properly, and not inadvertently skipping tests.

Each test consists of several subtests which check the instruction under test with various data patterns and ensure the accuracy of the condition codes.

The basic composition of a subtest is as follows:

1. Load GPR2 and GPR3, if needed, with the expected results.
2. Set up any operands, usually in GPR6 and GPR7, with test data.
3. Execute the instruction under test.

Following execution of the first pass, the program enables the system integrity traps and Real-Time Clock interrupt (RPL18).

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of Gould CONCEPT/32 CPU Diagnostic Unit - Part Two.

```
>LOAD CN.CP2.r
*LOADED: 'CN.CP2.r'
 PGM:  03000
>STAR CN.CP2.r PASS(100)

START CN.CP2.r (#111031-211r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

MODES:  EXTENDED MAPPED INTERVAL-TIMER
          NO       NO          NO
END PASS:          1
MODES:  EXTENDED MAPPED INTERVAL-TIMER
          YES      NO          NO
MODES:  EXTENDED MAPPED INTERVAL-TIMER
          YES      YES         NO
MODES:  EXTENDED MAPPED INTERVAL-TIMER
          YES      YES         YES
END PASS:        100

END CN.CP2.r:          100 PASS(ES)
```

Gould CONCEPT/32™

Central Processing Unit - Part Three

(CN.CP3.r)

Diagnostic Description

Diagnostic Number: 111031-212r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005520-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Central Processing Unit Diagnostic - Part Three (CN.CP3.r) Diagnostic Description Release 1.0, Publication Order Number **326-005520-000,** was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-2
4-1 through 4-2
5-1/5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2
3.3 Optional Programmed Halt 3-2

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test Structure 4-2

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 CPU Diagnostic-Part Three (CN.CP3.r) is a level one macrodiagnostic program that verifies the correct functioning of central processing unit (CPU) machine instructions.

The program consists of a sequence of tests which check the instructions in a logical manner, attempting not to use an instruction in a test sequence before verifying the instruction. An extensive condition code bit analysis is done on both the immediate result of the instruction under test and the result of the comparison of actual and expected data.

Options have been implemented that provide direct operator control over stop on error, loop on error, loop on test, and loop on subtest.

Note: This manual describes the specific attributes of CONCEPT/32 CPU Diagnostic - Part Three (CN.CP3.r). For a discussion of level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU, an IOP, and a minimum of 64K words of memory.

The program resides at a default starting address of 3000_{16}. The program must not be loaded into any memory module other than zero.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 1 to 10 seconds to execute each 100 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 CPU Diagnostic - Part One (CN.CP3.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial Program load (IPL) sequence and the load and start procedure, the program writes the following start message to the operater's console and begins execution:

```
START CN.CP3.r (#111031-212r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to Chapter 5.

2.2 Option Set

The following options are implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current mnessage before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subrtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Unimplemented.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Specify a pass count using the Diagnostic Overlay Loader START command.
7-8	Reset	Unimplemented.
9	Set	Traces all registers during test execution.
10	Set	Writes the pass count message to the operator's console upon completion of the first pass and every 100 passes thereafter.
11	Set	Writes the milestone messages to the Line printer.
12	Reset	Inhibits Extended Addressing. Perform extended addressing on every pass following pass one by resetting this option.
13	Reset	Inhibits Mapped Mode. Set mapped mode for every pass following pass two by resetting this option.
14	Reset	Inhibits the interval timer. Activate the interval timer for every pass following pass three by resetting this option.

Option	Default	Effect When Set
15-18	Reset	Unimplemented.
19	Reset	Echoes all Output to the line printer.
20	Reset	Unimplemented.
21	Reset	Writes a scratchpad image.
22-25	Reset	Unimplemented.
26	Reset	Causes the diagnostic to immediately upon error detection to allow for a snapshot of the error location. The program does not write and error message or save the contents of the GPRs. Nullify the halt by entering the Panel mode Run command on the operator's console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27-31	Reset	Reserved - Do not use.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 CPU Diagnostic - Part Three (CN.CP3.r), and the default option settings are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error message lists the pass count and the program counter location where the error occurred. In addition, the error message lists information specific to the error condition. Section 3.2 contains the standard format for error messages generated during the execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at a location immediately following the location where the error occurred. If in subsequent passes the error continues to occur, refer to the program counter location in the source code listing to determine the cause of the error condition.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program write out one or more of the following messages:

```
ERROR/PASS:xxxxxxxx, PC:xxxxx
EXP  CC's:xxxx   ANS:xxxxxxxx xxxxxxxx
ACT  CC's:xxxx   ANS:xxxxxxxx xxxxxxxx
```

ERROR/PASS — indicates the current program pass.

PC — indicates the program counter (PC) within the failing subtest.

EXP CC's — indicates software predicted (EXP) condition codes received from:

- the expected result of an instruction under test
- the expected result from a good comparison of data (usually CC=0001)

ANS — indicates the software predicted answer resulting from the instruction under test. Given in doubleword format, using GPR2 and GPR3, for simplicity of error reporting.

ACT CC's — indicates the actual received condition codes from:

- the result of executing the instruction under test.
- the result of the comparison of expected data and actual data.

ANS — indicates the actual resulting data from the instruction under test, as recorded in GPR6 and GPR7.

3.3 Optional Programmed Halt

If option 2 (halt on error) is set, the general purpose registers are loaded with execution status prior to the programmed Halt as follows:

- GPR2 contains current pass count
- GPR3 contains current error count
- GPR4 contains 60HZ clock count
- GPR5 contains arithmetic exception trap count
- GPR6 contains current test number in decimal

Option 2 does not affect GPRs 0, 1, and 7.

Entering the Panel mode Run command causes the program to reload the GPRs as they were at the time the error was detected, then halt program execution.

Entering the Panel mode Run command a second time causes the program to loop on error (if option 3 is set), or (if option 3 is reset) continue testing.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould CONCEPT/32 CPU Diagnostic - Part Three (CN.CP3.r) contains 46 tests which test the CPU Fixed Point Arithmetic instructions, Indexing, Indirect Addressing, and Format Codes for proper operation. This program also performs add, subtract, multiply, and divide tests using simulation routines. The instructions are tested in the following sequence:

- Add Register to Register
- Add Memory Word
- Add Memory Doubleword
- Add (Adder Tests)
- Add Register To Memory Doubleword
- Add Immediate
- Add Memory Byte
- Add Register To Memory Word
- Add Register To Memory Halfword
- Add Register To Memory Byte
- Add Register To Register Masked
- Add Memory Halfword
- Subtract Memory Word
- Subtract Register From Register
- Subtract Register From Register Masked
- Subtract Memory Byte
- Subtract Memory Halfword
- Subtract Immediate
- Subtract Memory Doubleword
- Subtract (Adder Tests)
- Compare Arithmetic Register to Register
- Compare Arithmetic With Memory Word
- Compare Arithmetic With Memory Doubleword
- Compare Arithmetic With Memory Halfword
- Compare Arithmetic With Memory Byte
- Compare Immediate
- Multiply Immediate
- Multiply By Memory Word
- Multiply Register By Register
- Multiply By Memory Byte
- Multiply By Memory Halfword
- Divide By Memory Byte
- Divide By Memory Halfword
- Divide By Memory Word
- Divide Register By Register
- Divide Immediate
- Indirect (1 Level)
- Format Code Tests
- Indirect (2 Level)
- Index (1 Level)
- Index (1 Level), Indirect (1 Level)
- Indirect (1 Level), Index (1 Level)
- Index (2 Levels), Indirect (1 Level)
- Index (3 Levels), Indirect (2 Levels)
- Multiply (R5) By (R6)
- Divide (R6, R7) by (R5)

Gould CONCEPT/32™

IOP Console

(CN.CON.r)

Diagnostic Description

Diagnostic Number: 111031-911r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005610-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

THE Gould CONCEPT/32 IOP Console (CN.CON.r) Diagnostic Description Release 1.0, Publication Order Number **326-005610-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vi
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-6
4-1 through 4-5/4-6
5-1 through 5-3/5-4

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2
2.2 CPU Execution Modes 2-3

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Test 4--Error Message 3-1
3.3 Test 5--Error Messages 3-1
3.4 Test 8--Error Message 3-1
3.5 Test 9--Test and Error Messages 3-2
3.5.1 Subtest 1--Check Loading of Memory Address (MA) 3-2
3.5.2 Subtest 2--Check Loading of Memory Data (MD) 3-2
3.5.3 Subtest 3--Check Reading and Writing of GPR 3-3
3.5.4 Subtest 4--Check Reading and Writing of PSW/PSD 3-3
3.5.5 Subtest 5--Check Reading and Writing of Control Switches 3-4
3.5.6 Subtest 6--Check Write Operand Stop 3-4
3.5.7 Subtest 7--Check Read Operand Stop 3-4
3.5.8 Subtest 8--Check Address Stop 3-4
3.5.9 Subtest 9--Check Instruction Stop 3-4
3.5.10 Subtest 10--Check Single Step 3-5
3.5.11 Subtest 11--Check Reading of Effective Address 3-5
3.5.12 Subtest 12--Check Panel Indicators 3-5
3.5.13 Subtest 13--Check Clock Override 3-5
3.5.14 Subtest 14--Check Primary/Secondary Panel (SECP/PRIP) 3-6
3.6 Unexpected Trap Messages 3-6

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test 1 - Interrupt Instruction Verification 4-1
4.3 Test 2 - Character Output 4-1
4.4 Test 3 - Keyboard/Printer Turnaround 4-2
4.5 Test 4 - Transfer Termination 4-2
4.6 Test 5 - Chaining Termination 4-2
4.7 Test 6 - Channel Error Conditions 4-3
4.8 Test 7 - Character Rotation 4-3
4.9 Test 8 - Modem Test 4-4
4.10 Test 9 - Control Panel Test 4-5

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 IOP Console Diagnostic (CN.CON.r) is a level one macrodiagnostic program that verifies the proper operation of the IOP console. The IOP console must be assigned to controller address $7E_{16}$.

The program reports errors by loading error information in the general purpose registers and executing an imbedded Halt instruction. Refer to the source code listing for comments relative to the cause of the error.

The program defaults the option word to allow for load and go execution without the need to modify the option word.

Select Port 0 and/or Port 1 of the IOP using the program's option word. Port 0 is the default.

Note: This manual describes the specific attributes of Gould CONCEPT/32 IOP Console Diagnostic (CN.CON.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

The program requires a CONCEPT/32 CPU with a minimum of 64K words of memory, and either a Model 8000 or a Model 8001 IOP.

The program resides at a default starting address of 3000_{16}. This program must not be loaded into any memory module other than zero.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

The program requires approximately 45 to 60 seconds to execute each pass depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 IOP Console Diagnostic (CN.CON.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console and begins execution:

```
START CN.CON.r (#111031-911r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

THE CONSOLE TERMINAL PRINTER PORT IS NOW ENABLED
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to chapter 5.

2.2 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0-1	Reset	Unimplemented.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be set using the Diagnostic Overlay Loader Start command.
7	Reset	Unimplemented.
8	Reset	Inhibits execution of all tests requiring operator intervention.
9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the operator's console upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Inhibits Extended Addressing mode. Perform extended addressing on every pass following pass one by resetting this option.
13	Reset	Inhibits Mapped mode. Set Mapped mode for every pass following pass two by resetting this option.
14	Reset	Inhibits interval timer. Activate the interval timer for every pass following pass three by setting this option.
15-16	Reset	Unimplemented.
17	Reset	Tests IOP Port 0 and 1.
18	Reset	Tests modem on Port 0.
19	Reset	Tests modem on Port 1.
20	Reset	Tests IOP Port 1 only.
21	Reset	Selects the control panel test (Test 9)
22-31	Reset	Reserved - Do not use.

2.3 CPU Execution Modes

This program tests all CPU execution modes. The following lists the execution mode under default conditions:

Pass #	Extended Mode	Mapped Mode	Interval Timer
1	No	No	No
2	Yes	No	No
3	Yes	Yes	No
4-n	Yes	Yes	Yes

During execution, the program writes a message to the operator's console each time the execution mode changes.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error occurs during the execution of Gould CONCEPT/32 IOP Console Diagnostic (CN.CON.r), and the default option settings are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at a location immediately following the location where the error occured. If in subsequent passes the error continues to occur, refer to the source code listing to determine the cause of the error.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

The following sections describe the error messages generated by this program.

3.2 Test 4--Error Message

The following message reports an error in terminating I/O to the console printer:

PRINTER TERMINATION UNSUCCESSFUL

3.3 Test 5--Error Messages

The following message reports an error in terminating data chained I/O commands:

DATA CHAINING TERMINATION UNSUCCESSFUL

The following message reports an error in terminating command chained I/O commands:

COMMAND CHAINING TERMINATION UNSUCCESSFUL

3.4 Test 8--Error Message

The following message reports an error in disconnecting the channel:

CHANNEL DISCONNECT FAILED

3.5 Test 9--Test and Error Messages

Any or all of the fifteen subtests of Test 9 can be performed on a single pass of the program. Some of these subtests result in a message being written to the operator's console for testing information or when an error is detected, as in subtest 2, for example. Other subtests rely on the operator to detect error conditions by comparing the expected data with the displayed data. The following subsections list the test messages written to the operator's console for each subtest and any possible error messages.

3.5.1 Subtest 1--Check Loading of Memory Address (MA)

```
TYPE      MA=0000688C(CR)
EXPECT    MA=0000688C        MD=0FFFFFFF
TYPE                         (CR)
EXPECT    MA=00006890        MD=00FFFFFF
TYPE                         (CR)
EXPECT    MA=00006894        MD=FF00FFFF
TYPE                         RUN(CR) TO CONTINUE
```

3.5.2 Subtest 2--Check Loading of Memory Data (MD)

```
TYPE      MA=000068EC(CR)
TYPE      MD=01234567(CR)
EXPECT    MA=000068EC        MD=01234567
TYPE      MD=89ABCDEF(CR)
EXPECT    MA=000068F0        MD=89ABCDEF
TYPE      (CR)
EXPECT    MA=000068F4        MD=89ABCDEF
TYPE      RUN(CR) TO CONTINUE
```

If an error is detected, the subtest writes the following message to the operator's console:

```
MEMORY DATA WRITE ERROR
INPUT DATA DOES NOT EQUAL EXPECTED DATA
```

3.5.3 Subtest 3--Check Reading and Writing of GPR

```
TYPE        GPR(CR)
EXPECT      GPR0=01234567
            GPR1=12345670
            GPR2=23456701
            GPR3=34567012
            GPR4=45670123
            GPR5=56701234
            GPR6=67012345
            GPR7=70123456

TYPE        GPR0=FFFFFFFF(CR)
            GPR1=11111111(CR)
            GPR2=22222222(CR)
            GPR3=33333333(CR)
            GPR4=44444444(CR)
            GPR5=55555555(CR)
            GPR6=66666666(CR)
            GPR7=77777777(CR)

TYPE        RUN(CR) TO CONTINUE
```

If an error is detected, the subtest writes the following message to the operator's console:

```
GENERAL PURPOSE REGISTER WRITE ERROR
INPUT DATA DOES NOT EQUAL EXPECTED DATA
```

3.5.4 Subtest 4--Check Reading and Writing of PSW/PSD

```
TYPE        PSW=F9004690(CR)
TYPE        PSD=00004000(CR)
TYPE        RUN(CR)
EXPECT      PSW=88004634          INST=EC0046B9

TYPE        PSD(CR)
EXPECT      PSD=88004634                =00000000

TYPE        RUN(CR) TO CONTINUE
```

If an error is detected, the subtest writes the following messages to the operator's console:

```
NO OPERATOR RESPONSE
NO (PSD) DATA RECEIVED

PROGRAM STATUS DOUBLEWORD 1 (PSW) WRITE ERROR
INPUT DATA DOES NOT EQUAL EXPECTED DATA

PROGRAM STATUS WORD 2 (PSD) WRITE ERROR
INPUT DATA DOES NOT EQUAL EXPECTED DATA
```

3.5.5 Subtest 5--Check Reading and Writing of Control Switches

```
TYPE        CS(CR)
EXPECT      MA=00000780          CS=00000000

TYPE        CS=FFF80000(CR)
EXPECT      MA=00000780          CS=FFF80000

TYPE        RUN(CR) TO CONTINUE
```

3.5.6 Subtest 6--Check Write Operand Stop

```
TYPE        WS=000068EC(CR)
TYPE        RUN(CR)
EXPECT      PSW=90004804         INST=00020002    WS  ST          HALT

TYPE        WS(CR) TO RESET WRITE OPERAND STOP

TYPE        RUN(CR) TO CONTINUE
```

3.5.7 Subtest 7--Check Read Operand Stop

```
TYPE        RS=000068EC(CR)
TYPE        RUN(CR)
EXPECT      PSW=880049FC         INST=00020002    RS  ST          HALT

TYPE        RS(CR) TO RESET READ OPERAND STOP

TYPE        RUN(CR) TO CONTINUE
```

3.5.8 Subtest 8--Check Address Stop

```
TYPE        AS=00004B1A(CR)
TYPE        RUN(CR)
EXPECT      PSW=90004B18         INST=00090002    AS  ST          HALT

TYPE        AS(CR) TO RESET ADDRESS STOP

TYPE        RUN(CR) TO CONTINUE
```

3.5.9 Subtest 9--Check Instruction Stop

```
TYPE        IS=00004C3A(CR)
TYPE        RUN(CR)
EXPECT      PSW=90004C38         INST=00090002    IS  ST          HALT

TYPE        IS(CR) TO RESET INSTRUCTION STOP

TYPE        RUN(CR) TO CONTINUE
```

3.5.10 Subtest 10--Check Single Step

```
TYPE        STEP(CR)
EXPECT      PSW=90004034        INST=2FF02FF0

TYPE        (CR)
EXPECT      PSW=90004036        INST=2FF02FF0

TYPE        RUN(CR) TO CONTINUE
```

3.5.11 Subtest 11--Check Reading of Effective Address

```
TYPE        EA(CR)
EXPECT      PSW=A0004E58        EA=000078EC

TYPE        STEP(CR)
TYPE        EA(CR)
EXPECT      PSW=88004E5C        EA=000088EC

TYPE        RUN(CR) TO CONTINUE
```

3.5.12 Subtest 12--Check Panel Indicators

```
TYPE        RUN(CR)
EXPECT                          RUN

TYPE        @@P
TYPE        HALT(CR)
EXPECT                  INT     WAIT                    HALT

TYPE        RST(CR)
EXPECT      PSW=80000000        INST=EC005029           HALT

TYPE        RUN(CR) TO CONTINUE
```

3.5.13 Subtest 13--Check Clock Override

```
TYPE        OVR(CR) TO SET CLOCK OVERRIDE
TYPE        RUN(CR) TO CONTINUE
TYPE        OVR(CR) TO RESET CLOCK OVERRIDE
TYPE        RUN(CR) TO CONTINUE
```

If an error is detected, the subtest writes the following message to the operator's console:

```
ERROR: xxxx RTC INTERRUPTS RECEIVED WITH OVR ON
```

xxxx specifies the hexadecimal number of interrupts received.

3.5.14 Subtest 14—Check Primary/Secondary Panel (SECP/PRIP)

```
MOVE TO SLAVE PANEL

TYPE      PRIP(CR) ON SLAVE PANEL
TYPE      RST(CR)
EXPECT    NO KEYBOARD ECHO, NO RESET
```

3.6 Unexpected Trap Messages

If option 2 (halt on error) is set, the program reports any unexpected traps by causing a programmed halt and writing the following messages to the operator's console:

```
      UNEXPECTED    description        TRAP
OLD PSD: xxxxxxxxxxxxxxxx        TRAP   ADDR: aaaa
PASS: cccccc
{NO EXTENDED ADDRESSING}      {UNMAPPED}{NO INTERVAL TIMER}
{EXTENDED ADDRESSING   }      {MAPPED  }{INTERVAL TIMER   }
```

description	specifies the type of unexpected trap detected by the program
xxx...xxx	specifies the value of the program status doubleword at the time the trap was detected
aaaa	specifies the address of the unexpected trap
cccccc	specifies the current pass count

The last line of the message identifies the CPU execution mode upon detection of an unexpected trap.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould IOP Console Diagnostic (CN.CON.r) contains nine tests designed to execute in sequence. Each test sequence verifies one or more channel controller functions, exceptional conditions (forced error conditions), or control panel functions.

During program execution, observe the correctness and quality of the written report including character clarity, character spacing, and cursor flight time. The program generates no programmed error halts during a normal execution sequence.

4.2 Test 1--Interrupt Instruction Verification

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 1 - INTERRUPT INSTRUCTION VERIFICATION

The test verifies that the basic Class F interrupt control instructions (Disable Channel (DCI), Enable Channel (ECI), Activate Channel (ACI), and Deactivate Channel (DACI)) perform correctly.

4.3 Test 2--Character Output

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 2 - CHARACTER OUTPUT

The test verifies the write transmission of the 64-character printable ASCII character set and produces the following output:

THE ALPHABET (A-Z) A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
THE NUMBERS (ZERO-NINE) 0 1 2 3 4 5 6 7 8 9
THE SPECIAL CHARACTERS

EXCLAMATION POINT	!	QUOTATION MARK	"
NUMBER SIGN	#	DOLLAR SIGN	$
PERCENT SIGN	%	AMPERSAND	&
APOSTROPHE	'	LEFT PARENTHESIS	(
RIGHT PARENTHESIS	)	ASTERISK	*
PLUS SIGN	+	COMMA	,
HYPHEN	-	PERIOD	.
SLASH	/	COLON	:
SEMICOLON	;	LESS THAN SIGN	<
EQUAL SIGN	=	GREATER THAN SIGN	>
QUESTION MARK	?	AT SIGN	@
LEFT BRACKET	[	RIGHT BRACKET	]
SPACE		REVERSE SLASH	\
BELL			

4.4 Test 3—Keyboard/Printer Turnaround

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 3 - KEYBOARD/PRINTER TURNAROUND

The test verifies the read transmission capability of the IOP and keyboard mechanism with auto character echoing performed in the channel, and the subsequent retransmission of the data back to the console. The keyboard read transfer is prompted by the following message:

..ENTER DATA AND TERMINATE WITH (cr)

Respond by keying one to 72 characters as desired and terminating the response with a carriage return. Failure to respond to the prompt within 10 seconds results in the following message:

..NO OPERATOR RESPONSE? (BELL)

To skip this test, set option 8 (inhibit operator intervention tests).

4.5 Test 4—Transfer Termination

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 4 - TRANSFER TERMINATION

The test verifies that the IOP Controller can correctly perform a terminate transfer function. The IOP Controller responds to a Halt I/O (HIO) instruction received while a data transfer is in progress by resetting the IOP Controller and generating a service interrupt. The program verifies the terminate transfer function in relation to a write to the console and a read from the console as indicated by the following messages:

..PRINTER TERMINATION
..KEYBOARD TERMINATION

4.6 Test 5—Chaining Termination

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 5 - CHAINING TERMINATION

The test verifies the proper termination of chained commands. The test terminates both command-chained and data-chained instructions using a Stop I/O (STPIO) instruction. The following message indicates proper execution of this test:

...DATA CHAINING TERMINATION
...COMMAND CHAINING TERMINATION

4.7 Test 6—Channel Error Conditions

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 6 - CHANNEL ERROR CONDITIONS

The test verifies that the IOP Controller correctly detects error conditions. The following conditions are checked:

- access to nonpresent memory to cause a channel program check
- back to back Start I/O instructions to cause a subchannel busy condition
- unit check by writing to the input subchannel
- unit check by reading to the output subchannel
- unit check by a Read Backwards command
- multiple interrupts received from back-to-back Start I/O and Halt I/O instructions issued to opposite subchannels

4.8 Test 7—Character Rotation

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 7 - CHARACTER ROTATION

The test provides character output of sufficient duration and line length to display the effects of cursor flight time, alignment and character spacing. The following is an example of the test results:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:; <=> ?@ [\]
BCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:; <=> ?@ [\]  A
CDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:; <=> ?@ [\]  AB
DEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:; <=> ?@ [\]  ABC
                                    .
                                    .
                                    .
] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:; <=> ?@ [\
 ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:; <=> ?@ [\]
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:; <=> ?@ [\]
```

4.9 Test 8--Modem Test

If option 11 (write milestone messages) is set, the following message identifies this test:

TEST 8 - MODEM TEST

The test verifies the ability of the console to perform its function via a full duplex modem link.

Options are provided to allow the modem test to be run using either one or both of the IOP Console I/O ports (Port 0 or Port 1). Setting option 18 (test modem on Port 0) enables the modem test for Port 0. Setting option 19 (test modem on Port 1) enables the modem test for Port 1.

It is advised that the modem test port not be the same as the port selected for the other console tests. For example, if Port 0 is the port selected as the console test port (this is the default port) set option 19 to test the modem on Port 1. Similarly, if option 20 (test IOP port only) is set to select Port 1 as the console test port, set option 18 to test the modem on Port 0. This procedure ensures that any errors in the modem test does not require the operator to re-establish the modem link in order to continue with the other tests.

To skip this test, set option 8 (inhibit operator intervention tests).

The modem test checks the connect, disconnect, and sense commands on both the nonmodem port and the modem port. The following message indicates normal execution:

CHANNEL DISCONNECT
CONNECT MODEM TO PORT 0 (1)

The operator has two minutes following the connect message to establish the modem connection. During this time, the program checks for the posting of an Attention status caused by the modem's ring-in.

Successful completion of the modem test causes the program to loop back to test 1 and run Tests 1 through 7 using the modem port as the console port.

Note: Test 8 executes only in the first pass in which it was selected. It does not automatically repeat if option 6 (repeat diagnostic) is set.

4.10 Test 9—Control Panel Test

If option 11 (write milestone messages) is set, the following message indentifies this test:

TEST 9 - CONTROL PANEL TEST

The test is composed of 14 independent subtests, each of which is designed to verify one or more control panel functions.

Each subtest executes in the following manner:

1. The subtest writes a message to the operator's console which identifies the subtest by name and number, instructs the operator to perform the control panel functions necessary for the test, and describes the expected test results.

2. The program halts, with the console in Panel mode, allowing the operator to proceed with the instructions. After performing each Panel mode operation, verify that the results were as predicted in the test message.

3. After completing the test instructions, enter the Panel mode Run command to continue the program. The program automatically starts the next subtest upon successful completion of each subtest. If the subtest detects an error, it writes an error message and executes a programmed halt. Enter the Panel mode Run command to continue the program after an error halt.

Once the program starts the control panel test, the execution sequence can be changed by setting the program counter (PC) to the start of any subtest and issuing the Panel mode Run command. Refer to the test directory in the source listing for the subtest addresses.

To skip this test, set option 8 (inhibit operator intervention tests).

Note: Test 9 executes only in the first pass after it is selected. It does not automatically repeat when option 6 is set.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of Gould CONCEPT/32 IOP Console Diagnostic (CN.CON.r).

```
>LOAD CN.CON.r
*LOADED: 'CN.CON.r'
 PGM: 03000
>STAR CN.CON.r PASS(0)

START CN.CON.r (#111031-911r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVSION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx EXTENDED ADDRESSING UNMAPPED NO INTERVAL TIMER

THE CONSOLE TERMINAL PRINTER PORT IS NOW ENABLED

TEST 1 - INTERRUPT INSTRUCTION VERIFICATION
TEST 2 - CHARACTER OUTPUT
THE ALPHABET (A-Z) A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
THE NUMBERS (ZERO-NINE) 0 1 2 3 4 5 6 7 8 9
THE SPECIAL CHARACTERS:
EXCLAMATION POINT       !     QUOTATION MARK       "
NUMBER SIGN             #     DOLLAR SIGN          $
PERCENT SIGN            %     AMPERSAND            &
APOSTROPHE              '     LEFT PARENTHESIS     (
RIGHT PARENTHESIS       )     ASTERISK             *
PLUS SIGN               +     COMMA                ,
HYPHEN                  -     PERIOD               .
SLASH                   /     COLON                :
SEMICOLON               ;     LESS THAN SIGN       <
EQUAL SIGN              =     GREATER THAN SIGN    >
QUESTION MARK           ?     AT SIGN              @
LEFT BRACKET            [     RIGHT BRACKET        ]
SPACE                         REVERSE SLASH        \
BELL

TEST 4 - TRANSFER TERMINATION
..PRINTER TERMINATION
..KEYBOARD TERMINATION
TEST 5 - CHAINING TERMINATION
..DATA CHAINING TERMINATION
..COMMAND CHAINING TERMINATION
TEST 6 - CHANNEL ERROR CONDITIONS
..CHECK INVALID MEMORY ACCESS
..CHECK PROGRAM VIOLATION
TEST 7 - CHARACTER ROTATION
```

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\]
BCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] A
CDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] AB
DEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABC
EFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCD
FGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDE
GHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEF
HIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFG
IJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGH
JKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHI
KLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJ
LMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJK
MNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKL
NOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLM
OPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMN
PQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNO
QRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOP
RSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQ
STUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQR
TUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRS
UVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRST
VWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTU
WXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUV
XYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVW
YZ0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWX
Z0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXY
0123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ
123456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0
23456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ01
3456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ012
456789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123
56789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ01234
6789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ012345
789 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
89 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567
9 !"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ012345678
!"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
!"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
"#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !
#$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"
$%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#
%&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$
&'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%
'()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&
()*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'
)*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'(
*+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()
+,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*
,-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+
-./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,
./:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-
/:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-.
:;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./
;<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:

```
<=>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;
= ?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<
>?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=
?@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>
@[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?
[\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@
\] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[
] ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\
 ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\]
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\]
```

THE CONSOLE TERMINAL PRINTER PORT IS NOW DISABLED

END CN.CON.r xxx PASS(ES)

Gould CONCEPT/32™

Control Store

(CN.CSD.r)

Diagnostic Description

Diagnostic Number: 111031-281r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005620-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Control Store (CN.CSD.r) Diagnostic Description Release 1.0, Publication Order Number **326-005620-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-5/2-6
3-1 through 3-2
4-1 through 4-4
5-1 through 5-3/5-4

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-3
2.3 CPU Execution Modes 2-5

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test 1 4-2
4.3 Tests 2 through 5 and Tests 45 through 48 4-2
4.4 Test 6 4-2
4.5 Test 7 4-2
4.6 Test 8 4-2
4.7 Tests 9 through 14 4-2
4.8 Tests 15 through 17, 20, 21, 28 and 29 4-2
4.9 Tests 18 and 19 4-3
4.10 Tests 22 through 27 4-3
4.11 Tests 30 through 43 4-3
4.12 Test 44 4-3
4.13 Test 49 4-3
4.14 Test 50 4-3
4.15 Test 51 4-4
4.16 Test 52 4-4
4.17 Test 53 4-4
4.18 Test 54 4-4
4.19 Test 55 4-4

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

The Gould CONCEPT/32 Control Store Diagnostic (CN.CSD.r) is a level one macrodiagnostic program that verifies the correct functioning of the Alterable Control Store (ACS) and the Writable Control Store (WCS) portions of the CONCEPT 32/67 processor.

Options have been implemented that provide direct operator control over test execution, milestone messages, error looping, and other functions.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Control Store Diagnostic (CN.CSD.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

The program requires a Gould CONCEPT 32/67 CPU, or both a CPU and an IPU, with a minimum of 64K words of memory.

The program resides at a default starting address of 3000_{16}. The program must not be loaded into any memory module other than zero.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 12 to 14 minutes to execute the first pass and 60 to 70 seconds to execute each subsequent pass depending upon the following:

- machine type
- machine configuration
- option setting

Note: These times are valid for a configuration of 12K of Control Store RAMs on one Gould CONCEPT 32/67 processor in PROM mode only. If two Gould CONCEPT 32/67 processors are configured on one machine, these times double.

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Control Store Diagnostic (CN.CSD.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

Note: A Control Store patch file name can be specified as a part of the Load command.

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start messages to the operator's console:

```
START CN.CSD.r (#111031-281r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

When a patch file is not specified in the Load command, the program writes the following message the operator's console:

```
NO PATCH FILE REQUESTED.  AT END WILL COPY PROMS
INTO ACS, AND ENABLE ACS MODE WITHOUT UPDATE
```

If a patch file is specified in the Load command, the program writes the following message to the operator's console:

```
(C67A3ACS) PATCH FILE FOUND.  AT END WILL COPY
PROMS INTO ACS, AND UPDATE FROM THE LOADED PATCH FILE
AND ENABLE ACS MODE
```

If the patch file specified is not resident, the program writes the following error message to the operator's console and returns control to the Diagnostic Overlay Loader:

```
CAN NOT FIND REQUESTED PATCH FILE (name)
```

The program then writes the following message, specifying the machine configuration, to the operator's console:

```
CPU TYPE = 32/xx
      WITH nK ACS
           nK WCS
           CPU WAS IN   { ROM-SIM }
                        { PROM    } MODE, NOW
                        { ACS     }

           RUNNING IN   { ROM-SIM }
                        { PROM    } MODE
                        { ACS     }
           FIRMWARE REVISION NO=rrrrrrrr rrrrrrrr
```

If an IPU is present, the program appends the following message:

```
IPU TYPE = 32/xx
      WITH nK ACS
           nK WCS
           IPU WAS IN   { ROM-SIM }
                        { PROM    } MODE, NOW
                        { ACS     }

           RUNNING IN   { ROM-SIM }
                        { PROM    } MODE
                        { ACS     }
           FIRMWARE REVISION NO=rrrrrrrr rrrrrrrr
```

xx specifies the Gould CONCEPT/32 machine type.

n specifies the size (in words) of available ACS/WCS.

r specifies the firmware revision number.

The program then writes one of the following messages, depending upon machine configuration, to the operator's console and execution begins:

```
WILL RUN TESTS IN xx CPU THEN xx IPU
WILL RUN TESTS IN xx CPU ONLY
WILL RUN TESTS IN xx IPU ONLY
ABORT!! M/C DOES NOT MEET MINIMUM CONFIGURATION
```

xx specifies the Gould CONCEPT/32 machine type.

2.2 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader Start command.
7	Reset	Enables clock interrupt.
8-9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the line printer upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Inhibits Extended Addressing mode. Perform extended addressing on every pass following pass one by resetting this option.
13	Reset	Inhibits Mapped mode. Set Mapped mode for every pass following pass two by resetting this option.
14	Reset	Inhibits interval timer. Activate the interval timer for every pass following pass three by setting this option.

Option	Default	Effect When Set
15	Reset	Inhibits all iterations. Perform a specified number of test iterations on every pass following pass four by resetting this option.
16	Reset	Inhibits all error message printouts.
17	Reset	Inhibits all error report printouts.
18	Reset	Unimplemented.
19	Reset	Echoes all messages to the line printer
20	Reset	Inhibits CPU/MM configuration printout.
21	Reset	Writes scratchpad image.
22	Reset	Forces the Interval Timer Clock mode. This option overrides option 14 when both are set.
23	Reset	Runs in a faster, but less complete mode.
24-25	Reset	Unimplemented.
26	Reset	Causes the program to immediately halt upon error detection to allow for a snapshot of the error location. The program does not write an error message or save the contents of the GPRs. Nullify the halt by entering the Panel mode Run command on the operator's console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27	Reset	Unimplemented
28	Reset	Executes tests in random order.
29	Reset	Executes tests in reverse order.
30	Reset	If option 31 is set, inhibits the repetitive error messages caused by simulated error conditions in iterative test sequences (i.e., the same test sequence repeated on various data configurations or memory locations). If option 31 is reset, this option is invalid.
31	Reset	Simulates all possible error conditions in the program tests to allow the operator to step through execution. This function may be used as a maintenance or debugging aid by verifying and demonstrating test operations and error message routines.

2.3 CPU Execution Modes

This program tests any or all Gould CONCEPT/32 CPU execution modes. The following lists the execution modes under default conditions:

Pass #	Extended Mode	Mapped Mode	Interval Timer	Iterations
1	No	No	No	No
2	Yes	No	No	No
3	Yes	Yes	No	No
4	Yes	Yes	Yes	No
5-n	Yes	Yes	Yes	Yes

During execution, the program writes a message to the operator's console each time the execution mode changes.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Control Store Diagnostic (CN.CSD.r), and the default options are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel made.

Error reporting consists of a message written to the operator's console describing the location of the error, a description of the error, a report of expected/actual data, possible erring traps, and program status. Refer to Section 3.2 for a description of the standard format for error messages generated during execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution of the program at a location immediately following the location where the error occured. If in subsequent passes the error continues to occur, refer to the source listing to determine the cause of the error.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program cause the programs to write out one or more of the following messages:

```
*********************************************************************
*IPU DETECTED/REPORTED ERROR *
*TEST  n., SUBTEST  n., ERROR nnnnn., PSD nnnnnnnn,nnnnnnnn
*PASS        n., ITERATION        n.
*
*TRAP PSD (PSD SAVED BY TRAP) IS INCORRECT
*
*EXP nnnnnnnn,nnnnnnnn
*ACT nnnnnnnn,nnnnnnnn
*XOR nnnnnnnn,nnnnnnnn
*TRAP TO LOC nnnnnnnn, FROM LOC nnnnnnnn
*
*OPTIONS REGISTER (LOC nnnn) = nnnnnnnn
*********************************************************************
```

TEST	indicates the failing test number.
SUBTEST	indicates the failing subtest number.
ERROR	indicates the error number.
PSD	indicates the Program Status Doubleword.
PASS	indicates the pass number where the error occurred.
ITERATION	indicates the iteration where the error occurred.
EXP	indicates the expected data.
ACT	incicates the data received.
XOR	indicates the results of an exclusive OR performed using the expected and actual data.
OPTIONS REGISTER	indicates the settings of the option register at the time of the error.
n	specifies either a hexadeciamal of decimal number. All numbers followed by a period are decimal numbers. Numbers that are not followed by a period are hexadecimal numbers.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

The Gould CONCEPT/32 Control Store Diagnostic (CN.CSD.r) level one macrodiagnostic program consists of 55 tests that verify the correct functioning of the Alterable Control Store (ACS) before the ACS is overlayed with firmware update patches. The tests are organized into groups, by function, as follows:

- Test 1 verifies the validity of the PROMs.
- Tests 2 through 5, and Tests 45 through 48 verify the ability to write on Control Store address boundaries.
- Tests 6 and 7 verify Control Store data lines.
- Test 8 verifies the ability to write to Control Store in PROM mode.
- Tests 9 through 14 verify the ability to read and write to main memory addresses using the WCS read and write instructions.
- Tests 15 through 17, Tests 20 and 21, and Tests 28 and 29 verify RAM data for all available Control Store locations.
- Tests 18 and 19 verify Control Store address lines.
- Tests 22 through 27 verify RAM addresses for all available Control Store locations.
- Tests 30 through 43 verify RAM noise for all available Control Store locations.
- Test 44 verifies the ability to switch to ACS.
- Test 49 verifies the ability to write to Control Store in ACS mode.
- Test 50 verifies the ability to jump on WCS boundaries.
- Test 51 verifies the ability to read, write, and jump on unprivileged WCS instructions.
- Test 52 verifies the ability to execute a WCS jump instruction in all available addressing modes.
- Test 53 verifies the ability to jump to all WCS addresses.
- Test 54 verifies the ability to execute a routine in WCS.
- Test 55 overlays ACS with firmware update patches.

4.2 Test 1

This test verifies that the data within the PROMs is valid and can be read using the RWCS instruction. The test ignores overflow.

4.3 Tests 2 through 5 and Tests 45 through 48

These tests verify the ability to write on Control Store address boundaries.

4.4 Test 6

This test verifies that all 64 Control Store data lines are able to achieve a high state and a low state from each 4K RAM bank of Control Store.

4.5 Test 7

This test verifies that none of the 64 Control Store data lines of a 4K RAM bank of Control Store is shorted.

4.6 Test 8

This test consists of three subtests that verify the ability to write to Control Store in PROM mode and enable writing to Control Store (when status bit 20 is set).

4.7 Tests 9 through 14

These tests verify the ability to read from and write to main memory addresses using the WCS read and write instructions under various error and nonerror conditions (i.e., map updates and invalid map on nonpresent memory).

4.8 Tests 15 through 17, 20, 21, 28 and 29

These tests verify the ability of each memory location in the Control Store RAMs to hold data.

4.9 Tests 18 and 19

These tests consist of three subtests that verify Control Store address lines.

4.10 Tests 22 through 27

These tests verify that each memory location in the Control Store RAMs can be uniquely addressed.

4.11 Tests 30 through 43

These tests verify RAM noise for all available Control Store locations.

4.12 Test 44

This test verifies the ability to switch to and execute firmware from ACS.

4.13 Test 49

This test consists of three subtests which verify the ability to write to Control Store in ACS mode and enable writing to WCS (when status bit 20 is set).

4.14 Test 50

This test verifies that the JWCS macro instruction executes only within the range of WCS addresses.

4.15 Test 51

This test verifies the ability to read, write, and jump on unprivileged WCS instructions.

4.16 Test 52

This test verifies the ability of the JWCS macro instruction to execute in all available addressing modes.

4.17 Test 53

This test verifies the ability of the JWCS macro instruction to jump to all WCS addresses and execute firmware instructions.

4.18 Test 54

This test verifies the ability to execute a routine in WCS.

4.19 Test 55

This test overlays the ACS RAMs with firmware update patches.

The test copies firmware from the PROMs into the ACS. If a patch file exists, the test overlays the patch file into the ACS and enters ACS mode.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of the Gould CONCEPT/32 Control Store Diagnostic (CN.CSD.r).

```
>LOAD CN.CSD.r
*LOADED: 'CN.CSD.r'
PGM:  03000
>STAR CN.CSD.r

START CN.CSD.r (#111031-281r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

(patchfle) PATCH FILE FOUND.  AT END WILL COPY PROMS INTO ACS
AND UPDATE FROM THE LOADED PATCH FILE

CPU TYPE = 32/67
        WITH      4K ACS
                  8K WCS
                  CPU WAS IN PROM      MODE, NOW
          RUNNING IN PROM MODE
          FIRMWARE REVISION NO= D60F5003 22904002

IPU TYPE = 32/67
        WITH      4K ACS
                  8K WCS
                  IPU WAS IN PROM      MODE, NOW
          RUNNING IN PROM MODE
          FIRMWARE REVISION NO= D60F5003 22904002

      WILL RUN TESTS ON xx CPU THEN xx IPU
      OPTIONS REGISTER (LOC 3044) = 22B00000

      MEMORY CONFIGURATION:
      (000000-1FFFFC)  ( 512.KW)   PRESENT
      (200000-FFFFFC)  (3584.KW) NON-PRESENT

         TEST01. PROM CHECKSUM TEST
         TEST02. CS ADDRESS BOUNDARIES. ST21=0, ST20=0, RS0=0
         TEST03. CS ADDRESS BOUNDARIES. ST21=0, ST20=0, RS0=1
         TEST04. CS ADDRESS BOUNDARIES. ST21=0, ST20=1, RS0=0
         TEST05. CS ADDRESS BOUNDARIES. ST21=0, ST20=1, RS0=1
```

```
TEST06. DATA LINE STUCK TESTS.
   SUBTEST: 1.
   SUBTEST: 2.
   SUBTEST: 3.
TEST07. DATA LINE SHORT TESTS.
   SUBTEST: 1.
   SUBTEST: 2.
   SUBTEST: 3.
TEST08. WRITE ENABLE TEST. PROM MODE.
   SUBTEST: 1.
   SUBTEST: 2.
   SUBTEST: 3.
TEST09. WWCS/RWCS WITH MEMORY C BITS SET.
TEST10. WWCS/RWCS WITH NON-PRESENT MEMORY.
TEST11. WWCS/RWCS WITH MAP UPDATE.
TEST12. WWCS/RWCS WITH INVALID MAP.
TEST13. WWCS/RWCS WITH PROTECTED MAP.
TEST14. WWCS/RWCS MAPPED TO NON-PRESENT MEMORY.
TEST15. CONTROL STORE ALL ZEROS.
TEST16. CONTROL STORE ALL ONES.
TEST17. CONTROL STORE CHECKERBOARD.
TEST18. ADDRESS LINE STUCK LOW TEST.
   SUBTEST: 1.
   SUBTEST: 2.
   SUBTEST: 3.
TEST19. ADDRESS LINE STUCK HIGH TEST.
   SUBTEST: 1.
   SUBTEST: 2.
   SUBTEST: 3.
TEST20. CONTROL STORE SLIDING ONES.
TEST21. CONTROL STORE SLIDING ZEROS.
TEST22. FULL ADDRESS LINE TEST.
   SUBTEST: 1.
          WORKING....0FF0/2FF0
   SUBTEST: 2.
          WORKING....1FF0/2FF0
   SUBTEST: 3.
          WORKING....2FF0/2FF0
TEST23. ADDRESS LINES MOD 3 PATTERN.
TEST24. ADDRESS LINES MOD 5 PATTERN.
TEST25. ADDRESS LINES MOD 7 PATTERN.
TEST26. ADDRESS LINES MOD 11 PATTERN.
TEST27. ADDRESS LINES MOD 13 PATTERN.
TEST28. REFRESH TEST ALL ZEROS.
TEST29. REFRESH TEST ALL ONES.
TEST30. MOVING INVERSIONS.  SEED=0001.
TEST31. MOVING INVERSIONS.  SEED=0010.
TEST32. MOVING INVERSIONS.  SEED=0011.
TEST33. MOVING INVERSIONS.  SEED=0100.
TEST34. MOVING INVERSIONS.  SEED=0101.
TEST35. MOVING INVERSIONS.  SEED=0110.
TEST36. MOVING INVERSIONS.  SEED=0111.
TEST37. MOVING INVERSIONS.  SEED=1000.
TEST38. MOVING INVERSIONS.  SEED=1001.
TEST39. MOVING INVERSIONS.  SEED=1010.
```

```
   TEST40.  MOVING INVERSIONS.  SEED=1011.
   TEST41.  MOVING INVERSIONS.  SEED=1100.
   TEST42.  MOVING INVERSIONS.  SEED=1101.
   TEST43.  MOVING INVERSIONS.  SEED=1110.
   TEST44.  SWITCH TO ACS MODE TEST.

*****ABOUT TO SWITCH TO ACS MODE*****


*****BACK IN PROM MODE*****

   TEST45.  CS ADDRESS BOUNDARIES.  ST21=1, ST20=0, RS0=0
   TEST46.  CS ADDRESS BOUNDARIES.  ST21=1, ST20=0, RS0=1
   TEST47.  CS ADDRESS BOUNDARIES.  ST21=1, ST20=1, RS0=0
   TEST48.  CS ADDRESS BOUNDARIES.  ST21=1, ST20=1, RS0=1
   TEST49.  WRITE ENABLE TEST. ACS MODE.
      SUBTEST: 1.
      SUBTEST: 2.
      SUBTEST: 3.
   TEST50.  WCS JUMP BOUNDARIES TEST.
   TEST51.  WWCS, RWCS, JWCS UNPRIVILEGED.
   TEST52.  JWCS DIRECT, INDIRECT, INDEXED, BASE, ETC.
   TEST53.  JWCS TO ALL WCS ADDRESSES.
   TEST54.  EXECUTE A WCS ROUTINE.
   TEST55.  FIRMWARE UPDATE.
PROMS COPIED INTO ACS RAMS, ENABLED
AND UPDATED TO F/W REV NO D60A5003 22904002
NOW EXECUTING FROM ACS RAMS


END PASS:

END CN.CSD.r            PASS(ES)
```

Gould CONCEPT/32™

Control Store Loader

(CN.CSL.r)

Diagnostic Description

Diagnostic Number: 111031-283r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005490-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Control Store Loader (CN.CSL.r) Diagnostic Description Release 1.0, Publication Order Number **326-005490-000,** was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-3/2-4
3-1/3-2
4-1/4-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Loading and Updating the Control Store 2-1
2.3 Milestone Messages 2-2

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-1

CHAPTER 4 EXAMPLE EXECUTION 4-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 Control Store Loader (CN.CSL.r) is a level one macrodiagnostic program that copies the firmware from the PROMs into the Alterable Control Store (ACS) on a 32/67 processor. The program also loads any current ACS patch file (if present) and updates the existing firmware. The loader can copy and update the firmware into both the CPU and the IPU of a CONCEPT 32/67 system.

Gould CONCEPT/32 Control Store Loader (CN.CSL.r) operates exclusively on the 32/67 processor. If started on any other processor, the program writes the start message to the operator's console followed by a message stating that the program is not valid on the current processor.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Control Store Loader Diagnostic (CN.CSL.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

The program requires the following:

- a CONCEPT 32/67 processor (a CPU, an IPU, or both)
- a minimum of 64K words of memory
- an optional patch file for updating existing firmware to the current revision level

The program resides at a default starting address of 3000_{16}. The program must not be loaded into any memory module other than zero.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Control Store Loader (CN.CSL.r) follows the load procedure established for level one macrodiagnostic programs. Because the program uses no options, the Diagnostic Overlay Loader Set, Reset, and Start commands need not be used. The following are the two steps to execute this program:

1. IPL the macrodiagnostic facility
2. Load the Control Store Loader program

Refer to Section 2.2 for a complete description of the load procedure for this program.

Following execution, the PROMs are copied into the Alterable Control Store (ACS) with no update.

Note: Loading the Alterable Control Store (ACS) without updating the existing CPU/IPU firmware can cause the program to report errors which are not indicative of hardware malfunctions.

2.2 Loading and Updating the Control Store

If specified on the Load command line, the Control Store patch file (67.ACx.r) must be loaded to update existing CPU/IPU firmware before executing any macrodiagnostic program. Following execution of the Control Store Loader program (CN.CSL.r), the PROMs are copied into the Alterable Control Store (ACS) with the updated firmware.

Specify the Control Store patch file (67.ACx.r) when loading the program. The following command line syntax specifies the Control Store patch file:

Syntax

DOL>LOAD CN.CSL.r @67.ACx.r;

CN.CSL.r — indicates that the Gould Control Store Loader program is to be loaded. Specify revision level by replacing the lower case letter r with the alphabetic character which represents the desired revision level. If no revison level is specified, the Diagnostic Overlay Loader loads the first copy of the program it finds.

@67.ACx.r; — indicates that the Control Store patch file is to be loaded. The lower case letter x must be replaced by either a 1, or a 2 specifying the processor revision. Specify revision level by replacing the lower case letter r with the alphabetic character which represents the desired revision level.

The above command line loads the Gould CONCEPT/32 Control Store Loader, copies the PROMs into ACS, and updates the CPU/IPU firmware.

2.3 Milestone Messages

Once loaded, the program writes its start message and one of the following messages to the operator's console and begins execution:

```
START CN.CSL.r (#111031-283r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

If no patch file is specified in the Load command, the program writes the following message:

```
NO PATCH FILE REQUESTED.  WILL COPY PROMS INTO ACS WITHOUT
UPDATE
```

If a patch file is specified using the Load command, the program writes the following message:

```
(67.ACx.r) PATCH FILE FOUND.  WILL COPY PROMS INTO ACS AND UPDATE
FROM THE LOAD PATCH FILE
```

If the patch file specified is not resident, the program writes the following error message and returns control to the Diagnostic Overlay Loader:

```
CAN NOT FIND REQUESTED PATCH FILE (xxxxxxxx)
```

Following the sequence of messages described above, the program writes a message specifying the machine configuration as follows:

```
CPU TYPE =  Vx
   WITH  nK ACS
         nK WCS

                    PROM
         RUNNING IN  ROM-SIM MODE
         FIRMWARE REVISION NO=D60F5003 22904002
```

If an IPU is present, the program appends the following message:

```
IPU TYPE =  32/xx
   WITH  nK ACS
         nK WCS

                    PROM
         RUNNING IN  ROM-SIM MODE
         FIRMWARE REVISION NO=D60F5003 22904002
```

xx indicates the CONCEPT/32 machine type.

n specifies the size (in words) of available ACS.

r specifies the firmware revision number.

Once the CPU firmware is copied into the ACS and updated by a Control Store patch file, the program writes the following message to the operator's console:

(filename) PATCHFILE FOUND. PROMS COPIED INTO ACS RAMS, ENABLED

AND UPDATED TO F/W REV NO rrrrrrrr rrrrrrrr

rrrrrrrr rrrrrrrr specifies the current revision number of the firmware update.

If an IPU is present, the program writes the following message to the operator's console indicating that control is being passed from the CPU to the IPU:

***** SWITCH TO IPU *****

Once the IPU firmware is copied into the ACS, the program writes the following message to the operator's console:

PROMS COPIED INTO ACS RAMS, ENABLED

If a patch file is loaded to update the IPU firmware, the program appends the following message indicating that the firmware update is complete, and control is passed from the IPU to the CPU:

AND UPDATED TO F/W REV NO rrrrrrrr rrrrrrrr
***** SWITCH TO CPU *****

At this point, the Control Store Loader is exited and control is returned to the Diagnostic Overlay Loader.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Control Store Loader (CN.CSL.r) the program writes an error message on the operator's console and returns control to the Diagnostic Overlay Loader.

The error message lists the error number, the Program Status Doubleword, and information specific to the error condition. Section 3.2 contains the standard format for error messages generated during the execution of this program.

To recover from an error condition, issue the Reload the program. If the program fails on successive execution attempts, refer to the source listing to determine the cause of the error.

3.2 Error Message Format

Errors generated during the execution of this program are preceded by the following header:

```
**********************************************************************
*ERROR xxxxx., PSD xxxxxxxx,xxxxxxxx
*
```

ERROR	indicates the error number.
PSD	indicates the Program Status Doubleword.

All numbers followed by a period are decimal. Numbers that are not followed by a period are hexadecimal.

Error messages which follow the above header relate specific error conditions which occur. If more than one error is generated, the program writes out the above header for each message generated.

CHAPTER 4

EXAMPLE EXECUTION

The following example displays a successful execution of the Control Store Loader program:

```
> LOAD CN.CSL.r
* LOADED: 'CN.CSL.r'
  PGM: 03000
> START CN.CSL.r PASS (1)

  START CN.CSL.r (#111031-283r date)
  (C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
  ALL RIGHTS RESERVED

  (67.ACx.r) PATCH FILE FOUND.  WILL COPY PROMS INTO ACS AND
  UPDATE FROM THE LOADED PATCH FILE

  CPU TYPE = 32/67
     WITH   4K ACS
       RUNNING IN PROM MODE
       FIRMWARE REV NO = C67A3003 22820003

  IPU TYPE = 32/67
     WITH   4K ACS
       RUNNING IN PROM MODE
       FIRMWARE REV NO = C67A3003 22820003

  WILL LOAD ACS IN 32/67 CPU THEN 32/67 IPU

  PROMS COPIED INTO ACS RAMS, ENABLED
     AND UPDATED TO F/W REV NO C67A3003 22820003

  *****SWITCH TO IPU*****

  PROMS COPIED INTO ACS RAMS, ENABLED
     AND UPDATED TO F/W REV NO C67A3003 22820003

  *****SWITCH TO CPU*****

  END PASS: 1
  END CN.CSL.r: 1 PASS(ES)
```

Gould CONCEPT/32™

Effective Address

(CN.EAD.r)

Diagnostic Description

Diagnostic Number: 111031-230r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005540-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Effective Address (CN.EAD.r) Diagnostic Description Release, 1.0 Publication Order Number **326-005540-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-3/3-4
4-1 through 4-2
5-1/5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2
2.3 CPU Execution Modes 2-3
2.4 Memory Configuration 2-3
2.5 Scratchpad Configuration 2-3
2.6 Map Register Configuration 2-3

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Header Information 3-2
3.3 Effective Address Trace 3-2
3.4 Test Result Reporting 3-3

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test Structure 4-2

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

The Gould CONCEPT/32 Effective Address Diagnostic (CN.EAD.r) is a level one macrodiagnostic program that verifies the effective address logic of memory-referencing instructions.

The program checks instructions under increasingly complex combinations of pre-indexed, indirect, and post-indexed addressing. The program then verifies the result of the instruction and addressing mode under test against the result of the same instruction using the equivalent direct address.

This program ensures that all instructions which reference memory behave consistently when given logically equivalent memory addresses. The program does not otherwise validate the results of these instructions.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Effective Address Diagnostic (CN.EAD.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU, an IOP, and a minimum of 64K words of memory.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 10 to 20 seconds to execute the first pass and 50 to 100 seconds for each subsequent 5 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Effective Address Diagnostic (CN.EAD.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to macrodiagnostic execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or rest options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following messages to the operator's console and begins execution:

```
START CN.EAD.r (#111031-230r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

Upon completion of the first pass, the program enables interrupts from the Real-Time Clock.

2.2 Option Set

The following options have been implemented for this program:

0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering RUN on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader Start command.
7-9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the line printer upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Causes the program to execute in unmapped, nonextended addressing mode only.
13	Reset	Causes the program to execute in unmapped, extended addressing mode only.
14	Reset	Causes the program to execute in mapped, nonextended addressing mode only.
15	Reset	Causes the program to execute in mapped, extended addressing mode only.
16-18	Reset	Unimplemented.
19	Reset	Echoes all output to the line printer.
20	Reset	Forces an effective address trace on each address under test.
21	Reset	Forces execution of the error handling routines on all tests.
22-31	Reset	Unimplemented.

2.3 CPU Execution Modes

This program tests any or all CONCEPT/32 CPU execution modes. Options 12 through 15 are used to select a single mode per pass as follows:

- Option 12 Set: Run unmapped, nonextended addressing mode only
- Option 13 Set: Run unmapped, extended addressing mode only
- Option 14 Set: Run mapped, nonextended addressing mode only
- Option 15 Set: Run mapped, extended addressing mode only

If any of the above options are set, the program executes a single set of tests in the selected mode only for each pass.

If options 12 through 15 are all reset, the program executes a single set of tests in all four modes in the order shown above for each pass.

If option 11 (write milestone messages) is set, the program writes the following message to the operator's console (or the line printer if option 19 is set) prior to the start of the automatic test sequence:

CPU MODE -- EXTENDED: {YES / NO} , MAPPED: {YES / NO}

2.4 Memory Configuration

The memory configuration required by the program consists of a minimum of 64K words. The program does not reference memory outside of the program area. The program can be offset loaded into any area of memory less than 128K words ($0\text{-}7FFFC_{16}$).

2.5 Scratchpad Configuration

The scratchpad configuration required by the program can contain any combination of device entries and interrupt entries as long as the IOP console and line printer controller (if desired) are configured as device addresses $7EFC_{16}$ and $7EF8_{16}$ respectively.

2.6 Map Register Configuration

The program constructs the map register configuration and loads it into the CPU map registers for program testing in mapped mode. The program marks each address in the first 128K words of the logical address space (map registers 00_{16} through $3F_{16}$) valid and mapped to the same address in the physical address space. Addresses beyond the first 128K words in the logical space are marked invalid.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Effective Address Diagnostic (CN.EAD.r) level one macrodiagnostic program, and the default options are set, the program writes out an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error message is organized into three areas:

- error header
- effective address trace message
- test results report

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at the location immediately following the location where the error occured. If in subsequent passes the error continues to occur, refer to the source listing to determine the cause of the error.

If the error condition fails to occur on subsequent passes of the program, restart the progam. If an error condition fails to occur after a restart, no error condition exists.

3.2 Header Information

The first four lines of the error message contain the following information:

```
TEST: xx  PSD: xxxxxxxx-xxxxxxxx
END PASS: xxxxx
CPU MODE -- EXTENDED: xxx, MAPPED: xxx
INSTRUCTION: xxxxxxxx x...x
```

TEST		indicates the number of the failing test.
PSD		indicates the program status doubleword immediately after the failing instruction.
END PASS		indicates the current program pass.
CPU MODE		
EXTENDED	xxx	YES indicates that the CPU is operating with extended addressing. NO indicates that the CPU is not operating with extended addressing.
MAPPED	xxx	YES indicates that the CPU is operating with memory mapping. NO indicates that the CPU is not operating with mapped memory.
INSTRUCTION		indicates the hexadecimal representation of the failing instruction, followed by the name of the instruction.

3.3 Effective Address Trace

The following lines contain a table tracing the calculation of effective address for the address under test. Each line describes one address level in the indirect chain. The first line always describes the address contained in the instruction word. Additional lines, if any, describe each of the indirect address words.

IX	I	F	ADDR	INDEX	EFFECTIVE:	F	WORD-ADDR	C0	C1
x	x	x	xxxxx	xxxxxxxx		x	xxxxxx	x	x
x	x	x	xxxxx	xxxxxxxx		x	xxxxxx	x	x
x	x	x	xxxxx	xxxxxxxx		x	xxxxxx	x	x

IX	if present, indicates the number (1, 2, or 3) of the index register; if absent, indicates that indexing is not used at this level.
I	indicates the indirect field (0 or 1) of the address word--it will always be 0 in the last line of the trace, and 1 in all other lines.
F	indicates the F bit (0 or 1) in the address word.
ADDR	indicates the 19 bit address field of the address word.
EFFECTIVE:	The next four columns indicate the effective F bit, Word Address, and two C bits, respectively, at this level of the indirect chain. If the I bit on this line is set, then the effective Word Address is the address of the next indirect word. The effective address of the address chain is shown by the effective F bit, C bits, and Address Word in the last line of the trace table.

The program does not print the Effective Address Trace if the trace of the same address chain has already been printed in the error message of another instruction.

3.4 Test Result Reporting

The next four lines report the simulated and actual result of executing the failing instruction.

```
            CC'S      R6          R7        MEMORY LOC xxxxxx
 INITIAL    xxxx   xxxxxxxx-xxxxxxxx      xxxxxxxx-xxxxxxxx
EXPECTED    xxxx   xxxxxxxx-xxxxxxxx      xxxxxxxx-xxxxxxxx
  ACTUAL    xxxx   xxxxxxxx-xxxxxxxx      xxxxxxxx-xxxxxxxx
```

MEMORY LOC — indicates the starting address of the two word memory location containing the target of the instruction.

INITIAL — indicates the contents of the Condition Codes, General Purpose Registers 6 and 7 (GPR6 and GPR7), and the memory target just prior to execution of the failing instruction.

EXPECTED — If no trap is expected, this line indicates the simulated Condition Code, GPR6 and GPR7, and memory target contents which should result from execution of the instruction. If a trap is expected, this line will contain the following message:

```
**** xxxx TRAP ****
```

xxxx — indicates any one of the following traps:

```
 NPM -- Nonpresent Memory Trap
 UNI -- Undefined Instruction Trap
PRIV -- Privilege Violation Trap
 MAP -- Map Fault Trap
ADRS -- Address Specification Trap
```

ACTUAL — If no trap occurs on execution of the failing instruction, this line indicates the Condition Code, GPR6 and GPR7, and the memory target contents after execution. If a trap occurs, the line contains one of the trap messages described above.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

The Gould CONCEPT/32 Effective Address Diagnostic (CN.EAD.r) level one macrodiagnostic program consists of thirteen tests. Each test verifies a different combination of indirect and indexed addressing. They are as follows:

Test 1:	Indexed direct addressing
Test 2:	Indirect addressing
Test 3:	Pre-indexed indirect addressing
Test 4:	Post-indexed indirect addressing
Test 5:	Pre-indexed indirect post-indexed addressing
Test 6:	Indirect to indirect addressing
Test 7:	Pre-indexed indirect to indirect addressing
Test 8:	Post-indexed indirect to indirect addressing
Test 9:	Pre-indexed indirect post-indexed to indirect addressing
Test 10:	Indirect to indirect post-indexed addressing
Test 11:	Pre-indexed indirect to indirect post-indexed addressing
Test 12:	Post-indexed indirect to indirect post-indexed addressing
Test 13:	Pre-indexed indirect post-indexed to indirect post-indexed addressing

4.2 Test Structure

Each test sets up the effective address chain to be tested and branches to a common subroutine. The subroutine varies the F and C bits at the different address levels and tests each combination with all memory reference instructions. Those instructions which require general purpose registers use GPR6 and/or GPR7.

Certain control instructions require special techniques to determine which effective address they are using. For LOAD PSD instructions, the two word memory target contains two PSD-WORD-1s, the first with all Condition Codes set, and the second with all CCs reset. For the Execute Memory instruction, the target contains the following instructions:

```
MEM.TARG ABR   6,0
         ABR   6,8
         ABR   6,16
         ABR   6,24
```

For Branch instructions, the target contains the following instructions and is immediately followed by a Branch returning control to the program:

```
MEM.TARG SLC   7,4
         SLC   7,4
         SLC   7,4
         SLC   7,4
```

Before testing an instruction, the program loads GPR6 and GPR7, sets the CCs to their initial condition, and loads the two word memory location containing the target of the effective address.

The program executes each instruction under test twice per effective address. The first time, the program executes the instruction with the addressing mode being tested. The program saves the resulting Condition codes, GPR6 and GPR7, memory target, and trap (if any) as the actual result.

On the second execution, the program executes with a direct address having the same effective address. The program saves the resulting CCs, GPR6 and GPR7, memory target, and trap (if any), possibly modified, as the simulated, or expected result. (Modification of the expected result is necessary for those instructions which give different results for direct and indirect addressing.)

An instruction passes the test for the addressing mode if it meets either of the following conditions:

- the same trap occurred as part of the expected and actual results
- no trap occurred as either the expected or the actual result, and the expected and actual CCs, GPR6 and GPR7, and memory target contents are the same

If the above conditions for passing the test are not met, the program reports this through its error reporting routine.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of the Gould CONCEPT/32 Effective Address Diagnostic (CN.EAD.r) level one macrodiagnostic program.

```
>LOAD CN.EAD.r
*LOADED: 'CN.EAD.r'
 PGM: 03000
>STAR CN.EAD.r PASS(1)

START CN.EAD.r (#111031-230r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED


CPU TYPE = 32/xx

CPU MODE --  EXTENDED: NO, MAPPED: NO
TEST  1    --  INDEX
TEST  2    --          INDIRECT
TEST  3    --  INDEX   INDIRECT
TEST  4    --          INDIRECT   INDEX
TEST  5    --  INDEX   INDIRECT   INDEX
TEST  6    --          INDIRECT               INDIRECT
TEST  7    --  INDEX   INDIRECT               INDIRECT
TEST  8    --          INDIRECT   INDEX       INDIRECT
TEST  9    --  INDEX   INDIRECT   INDEX       INDIRECT
TEST 10    --          INDIRECT               INDIRECT   INDEX
TEST 11    --  INDEX   INDIRECT               INDIRECT   INDEX
TEST 12    --          INDIRECT   INDEX       INDIRECT   INDEX
TEST 13    --  INDEX   INDIRECT   INDEX       INDIRECT   INDEX

END PASS:       1


END CN.EAD.r:           1 PASS(ES)
```

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Interrupt (CN.INT.r) Diagnostic Description Release 1.0, Publication Order Number **326-005560-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-3/3-4
4-1 through 4-6
5-1 through 5-4

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTRER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2
2.3 CPU Execution Modes 2-3
2.4 Execution Map Configuration 2-3

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2
3.3 Optional Programmed Halt 3-3

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test 1--Scratchpad Memory Checkout 4-1
4.3 Test 2--Basic Instruction Checkout 4-3
4.4 Test 3--Interrupt and Trap Configuration 4-3
4.5 Test 4--Instruction Checkout--All Levels 4-5
4.6 Test 5--Level Context Switching 4-5
4.7 Test 6--Active Inhibit 4-5
4.8 Test 7--Request in Interrupt Routine 4-5
4.9 Test 8--Active Stack 4-5
4.10 Test 9--Request Stack 4-5
4.11 Test 10--External Interrupt Checkout 4-6

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

The Gould CONCEPT/32 Interrupt Diagnostic (CN.INT.r) is a level one macrodiagnostic program that locates, by use of the scratchpad, and tests all priority interrupts on either the IOP or the Real-Time Option Module (RTOM). Each test verifies a unique property of an interrupt instruction, its address generation, or its priority queuing.

Options have been implemented that provide direct operator control over tests to be executed, milestone messages, error looping, and other functions.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Interrupt Diagnostic (CN.INT.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU with a minimum of 64K words of memory and either of the following:

- One to six Model 2345 Real-Time Option Modules
- One IOP Model 8000 or 8001.

The program resides at a default starting address of 3000_{16}. This program must not be loaded into any memory module other than zero.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 5 to 20 seconds to execute the first pass and 10 to 30 seconds to execute each subsequent 10 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Interrupt Diagnostic (CN.INT.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following messages to the operator's console and begins execution:

```
START CN.INT.r (#111031-240r  date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution, refer to Chapter 5.

2.2 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering RUN on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader Start command.
7-9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the line printer upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Inhibits Extended Addressing mode. Perform extended addressing on every pass following pass one by resetting this option.
13	Reset	Inhibits Mapped mode. Set Mapped mode for every pass following pass two by resetting this option.
14-16	Reset	Unimplemented.
17	Reset	Writes map register dump.
18	Reset	Unimplemented.
19	Reset	Echoes all output to the line printer.
20	Reset	Executes test 10. This test requires the manipulation of an external stimulus cable.
21	Reset	Writes scratchpad image.
22-31	Reset	Unimplemented.

2.3 CPU Execution Modes

This program tests any or all CONCEPT/32 CPU execution modes. The following lists the execution modes under default conditions:

Pass #	Extended Addressing	Mapped Mode
1	No	No
2	Yes	No
3-n	Yes	Yes

2.4 Execution Map Configuration

The program begins execution with memory module zero (map registers 00-04) marked valid. All other map entries are marked invalid and nonpresent ($07FF_{16}$). All tests run in the Mapped mode use this as the execution map configuration. The execution map is confirmed with all map registers in a one-to-one mapping relationship with all map blocks.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Interrupt Diagnostic (CN.INT.r), and the default options are set, the program writes out an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error message indicates the number, priority level, and addresses of traps or interrupts received. Refer to Section 3.2 for a description of the error message format.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at the location immediately following the location where the error occured. If in subsequent passes the error continues to occur, refer to the source listing to determine the cause of the error.

If the error condition fails to occur on subsequent passes of the program, restart the program. If an error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program are written in the following format:

{NO / ONE / xxxx} {INTRPT(S) / TRAP(S)} REC FROM Plyy AT aaaaa, {Plzz EXP / EXP AT bbbbb / Plzz EXP AT bbbbb / NO {INTRPT(S) / TRAP(S)} EXP}

NO	indicates that no interrupt or trap is received.
ONE	indicates that one and only one interrupt or trap is received.
xxxx	specifies the number of interrupts or traps received if more than one interrupt or trap is received.
Plyy	indicates either the priority level of the received interrupt or trap if ONE or xxxx interrupts are received. If no interrupt or trap is received, this output indicates the priority level of the expected interrupt or trap.
aaaaa	specifies the actual address at the point of interrupt or trap if one or more interrupts or traps are received. If no interrupt or trap is received, but one is expected, this output specifies the expected address at the point of interrupt or trap.
Plzz	indicates the priority level of the interrupt or trap expected.
bbbbb	specifies the expected address at the point of the interrupt or trap if an interrupt or trap is received and aaaaa does not equal bbbbb.
NO {INTRPT(S) / TRAP(S)} EXP	indicates that no interrupt or trap are expected.

Note: All addresses are expressed as program listing vectors.

3.3 Optional Programmed Halt

If option 2 (halt on error) is set, the program formats the general purpose registers as follows prior to the programmed halt:

- GPR0 Memory relative address of the call to the CHECKXX routine that decodes the error.
- GPR1 Number of interrupts received on a single request.
- GPR2 Defined by the test sequence.
- GPR4 Listing vector address of the call to the CHECKXX routine that decoded the error.
- GPR6 Contains the values for the priority level (bits 0-7) and the interrupt/trap address (bits 13-31) for an expected interrupt/trap.
- GPR7 Contains the actual hardware values for the priority level (bits 0-7) and the interrupt/trap address (bits 13-31) for an interrupt/trap received.

Option 2 does not affect GPR 3 and 5.

Rules for Optional Programmed Halt

- If the error associated with the failure indicates that no interrupt was received and GPR1 equals zero, the contents of GPR7 contains information pertaining to the last interrupt serviced.
- Any error message not covered above may be found in the description of the particular test to which it pertains.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould CONCEPT/32 Interrupt Diagnostic (CN.INT.r) level one macrodiagnostic program consists of 10 tests that test all priority interrupts on either the IOP or the Real-Time Option Module (RTOM). The program executes Tests 1 through 9 automatically by default. Execute Test 10 by setting option 20 (execute Test 10).

4.2 Test 1--Scratchpad Memory Checkout

This test verifies proper operation of the data paths to and from the CPU scratchpad memory and the unique addressability of each location. A copy of the initial scratchpad configuration is stored in a buffer until the test is completed. The program rolls in the scratchpad image for the remainder of the pass. The program executes this test on pass one.

Upon entering this test, the program writes the following message to the operator's console:

..CAUTION - SCRATCHPAD TEST IN PROGRESS -
- DO NOT TOUCH CONSOLE -

Failure to comply can result in a total system crash.

After the program rolls the scratchpad out to memory, the program tests the scratchpad for data path integrity and cross talk by using all zeros, all ones, walking ones, and walking zeros data patterns.

To verify the unique addressability of each scratchpad cell, the program performs the following two tests:

1. The first test uses a background pattern of all ones and stores that pattern in each scratchpad address (locations 00_{16} through FF_{16}).

2. The second test uses a background of all zeros and stores that pattern in each scratchpad address (locations FF_{16} through 00_{16}).

The following is an example of a system scratchpad image obtained by setting option 21 (write scratchpad image). The program writes the image to the operator's console by default, or to the line printer if option 19 (write messages to line printer) is set.

System Scratchpad Image

ADDR:	SCRATCHPAD DATA:			
00-03	0E7B0100	0E7B0101	0E7B0102	0F7B0100
04-07	0E7A0400	0E7A0401	0E7A0402	0F7A0400
08-0B	0E790800	0E790801	0E790802	0F790800
0C-0F	0E780C00	0E780C01	0E780C02	0F780C00
10-13	0F771000	0F771000	0F771000	0F771000
14-17	0F771000	0F771000	0F771000	0F771000
18-1B	0E761800	0E761801	0E761802	0E761803
1C-1F	0E761804	0E761805	0E761806	0E761807
.				
.				
6C-6F	0E71600C	0E71600D	0E71600E	0E71600F
70-73	0E707000	0E707001	0E707002	0E707003
74-77	0E707004	0E707005	0E707006	0E707007
78-7B	816F7800	00000000	806E7802	00000000
7C-7F	0F6D7C00	03527D04	0F6C7E00	03647F04
80-83	00FF7F0F	00FE7F0E	00FD7F0D	00FC7F0C
84-87	0E7B0100	0E7A0400	0E790800	0F780C00
88-8B	0F771000	0E761800	0E752000	0F743000
8C-8F	0E734000	0E725000	0E716000	0F707000
90-93	816F7800	806E7802	0F6D7C00	0F6C7E00
94-97	00EB7F0B	00EA7F0A	00E97F09	00E87F08
98-9B	00E77F06	00E67F07	00E57F05	03647F04
9C-9F	00E37F03	00E27F02	00E17F01	00E07F00
A0-A3	00DF7D0F	00DE7D0E	00DD7D0D	00DC7D0C
A4-A7	00DB7D0B	00DA7D0A	00D97D09	00D87D08
A8-AB	00D77D07	00D67D06	00D57D05	00D47D04
AC-AF	00D37D03	00D27D02	00D17D01	00D07D00
.				
.				
THROUGH				
.				
.				
F0-F3	00000080	00000100	00000700	00000788
F4-F7	00007800	00000000	00000000	ECDAB897
F8-FB	00000000	00000042	007F0000	00000000
FC-FF	00000000	00000000	00000000	00000000

4.3 Test 2--Basic Instruction Checkout

This test verifies integrity in the basic interrupt functions through the execution of various combinations of the interrupt instructions--Disable (DI), Enable (EI), Request (RI), Activate (AI), and Deactivate (DAI). Through the execution of 16 subtests, priority level 03 is verified in relation to itself, and in relation to priority levels 01 and 02.

4.4 Test 3--Interrupt and Trap Configuration

This test determines the interrupt and trap configuration present by directing Request Interrupt (RI) instructions at all priority levels $00\text{-}5F_{16}$. A priority level is defined as being present if an interrupt is received.

While scanning all priority levels, the program checks to determine if any of the following conditions are present:

- An interrupt level generates dynamic requests which are received from an enabled level before a request is issued.
- An interrupt is received from a priority level whose dispatch linkage queue vector indicates that a nonstandard dedicated location was used.
- An interrupt is received that is not the same as requested.

If any of the above conditions exist, the program disables the priority level and writes a message indicating that the priority level was skipped and the reason. The program removes the level from the list of available priority levels.

If the level received is different from the level requested, it could indicate an error in the scratchpad configuration, or that more than one interrupt is connected to a device.

Once the program determines that all levels are present, the program proceeds to assign priorities to all levels. The net result of the test is a log of all priority levels present in order of priority with an indication of priority levels skipped. Only those priority levels reported are tested by the program.

If a dynamic request occurs, the program appends the following message to the priority level message:

,SKIPPED (APPARENTLY DYNAMIC)

If a requested interrupt level vectors through a nonstandard dedicated location, the program appends the following message to the priority level message:

,SKIPPED (NON-STD DED-L)

The following is an example configuration of traps and interrupts from a small system with an IOP Controller. The program writes a milestone message in this format unless option 11 (write milestone messages) is reset.

Levels Present (By Priority)

PL	DED-L	LNK-L	TYPE	CLASS	ADDR
00	00080	2100	TRAP		
01	00084	2120	TRAP		
02	00088	2160	TRAP		
03	0008C	2180	TRAP		
04	00090	21A0	TRAP		
05	00094	21C0	TRAP		
06	00098	5478	TRAP		
07	0009C	2220	TRAP		
08	000A0	2240	TRAP		
09	000A4	2260	TRAP		
0A	000A8	2200	TRAP		
0B	000AC	2310	TRAP		
0C	000B0	22F0	TRAP		
0D	000B4	2200	TRAP		
0E	000B8	2280	TRAP		
0F	000BC	22A8	TRAP		
00	00100	6DBC	EXT		
01	00104	6DE0	EXT		
02	00108	6E04	EXT		
03	0010C	6E28	EXT		
10	00140	6FFC	I/O		
11	00144	7020	I/O		
13	0014C	5090	I/O	F 7E	
14	00150	708C	EXT		
15	00154	70B0	EXT		
16	00158	7004	EXT		
17	0015C	70F8	EXT		
18	00160	711C	EXT		
19	00164	7140	EXT		
1A	00168	7164	EXT		
1B	0016C	7188	I/O	3 7F	
1C	00170	71AC	EXT		
1D	00174	71D0	EXT		
1E	00178	71F4	EXT		
1F	0017C	7218	EXT		

DED-L — indicates the priority level's System Reset dedicated location.

LNK-L — indicates the address of the nine-word interrupt context block used to field the interrupt or trap.

TYPE — indicates either a trap, RTOM interrupt, or I/O interrupt.

CLASS and ADDR — indicate the data contained in the scratchpad entry for I/O devices including Class F devices.

4.5 Test 4—Instruction Checkout--All Levels

This test establishes the same degree of integrity for all available priority levels that Test 1 establishes on priority level 03. Through the execution of 15 subtests, each available priority level is verified in relation to itself, and in relation to priority levels 01 and 02.

The complete test is executed twice. The first pass allows the common interrupt receiver INT3 to exit using a Deactivate Interrupt (DAI)/Branch Indirect instruction sequence. The second pass allows the common interrupt receiver INT1 to exit using a Branch and Reset Interrupt Indirect instruction sequence.

4.6 Test 5—Level Context Switching

This test verifies the ability of the interrupt logic to perform a two-level controlled context switch.

4.7 Test 6—Active Inhibit

This test verifies that, when active, a senior priority level cannot be interrupted by a lower priority level.

4.8 Test 7—Request in Interrupt Routine

This test verifies that if a priority level is active by request, no lower priority level can interrupt. All higher priority levels can interrupt.

4.9 Test 8—Active Stack

This test verifies the ability of active and request queues to intermesh and selectively service all available levels.

4.10 Test 9—Request Stack

This test verifies the ability of the request queue to unload serially by priority when all available levels are pending by request.

4.11 Test 10--External Interrupt Checkout

This test verifies that all interrupt levels are available, and up to four input/output processors or five Real-Time Option Modules (0 through 4) are capable of responding to an external stimulus. Levels tested are: 00-03, and 14-5F. All service interrupts are excluded.

The options function as follows:

- options 3 and 4 allow for looping on a single interrupt level
- option 5 allows looping on a single IOP/RTOM
- jumpers may be required for signals LDVINTR04, 09, and 11 prior to starting this test (consult the RTOM Drawings Manual for jumper locations)

This test is implemented with a external stimulus cable (part number 144-103116-001 for a RTOM, 144-103425 for an IOP). This allows testing of an interrupt level stimulated with an external request, and also the ability of an interrupt request to be transmitted to an external element (possibly an RTOM on another SelBUS for synchronous interrupt functions).

The test checks sixteen external interrupt levels on each RTOM, while four external interrupts are tested on IOP's. Successful completion of the test is indicated by no error messages and a message to continue testing at the next available IOP/RTOM.

Note: This test cannot differentiate between the failure of a level under test to respond to external stimulus and the ability of the cross-connected interrupt to transmit the request.

This test executes only if option 20 (execute Test 10) is set. When it is entered, the test writes the following messages to the operator's console:

```
TURN CLOCK OVERRIDE ON
ATTACH EXTERNAL STIMULUS CABLE TO IOP/RTOM #
TO EXIT TEST 10-RESET OPTION 20
PROGRAM HALTED, 'RUN' TO CONTINUE
```

The clock override must be turned on to prevent spurious requests while PL18 is enabled.

To complete this test the following steps must be followed:

1. Do not power down, but attach the cross-connect cable (P1A to P1C on a RTOM, P3 on an IOP) indicated in the prompt message.
2. Enter RUN on the IOP console.

The algorithm for this test is as follows:

1. Enable level under test.
2. Disable complement level in IOP/RTOM under test.
3. Request complement level.
4. Wrap around request to level under test.
5. Fire level under test.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of the Gould CONCEPT/32 Interrupt Diagnostic (CN.INT.r).

```
> LOAD CN.INT.r
 *LOADED: 'CN.INT.r'
  PGM:  03000
> STAR CN.INT.r PASS(10)

START CN.INT.r (#111031-240r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

        MEMORY CONFIGURATION:
        (000000-1FFFFC)   ( 512KW)          PRESENT
        (200000-FFFFFC)   (3584KW)      NON-PRESENT
MODES:  EXTENDED MAPPED
              NO     NO
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
..CAUTION - SCRATCHPAD TEST IN PROGRESS -
            - DO NOT TOUCH CONSOLE -
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
LEVELS PRESENT (BY PRIORITY):
PL  DED-L  LNK-L  TYPE  CLASS ADDR
00  00080  03100  TRAP
01  00084  03120  TRAP
02  00088  03160  TRAP
03  0008C  03190  TRAP
04  00090  031C0  TRAP
05  00094  031F0  TRAP
06  00098  088F4  TRAP
07  0009C  03280  TRAP
08  000A0  032B0  TRAP
09  000A4  032E0  TRAP
0A  000A8  03250  TRAP
0B  000AC  033C0  TRAP
```

```
0C  000B0   03390  TRAP
0D  000B4   03370  TRAP
0E  000B8   03310  TRAP
0F  000BC   03340  TRAP
00  00100   09FC4  EXT
01  00104   09FF0  EXT
02  00108   0A01C  EXT
03  0010C   0A048  EXT
06  00118   0A0CC  I/O    F              08
08  00120   0A124  I/O    F              10
06  00118   0A0CC  I/O    F              08
08  00120   0A124  I/O    F              10
13  0014C   06F48  I/0    F              7E
14  00150   0A334  EXT
15  00154   0A360  EXT
16  00158   0A38C  EXT
17  0015C   0A3B8  EXT
18  00160   0A3E4  EXT
19  00164   0A410  EXT
1A  00168   0A43C  EXT
1B  0016C   0A468  I/O    3              7F
1C  00170   0A494  EXT
1D  00174   0A4C0  EXT
1E  00178   0A4EC  EXT
1F  0017C   0A518  EXT
TEST  4 - INSTRUCTION CHECKOUT
TEST  5 - LEVEL CONTEXT SWITCHING
TEST  6 - ACTIVE INHIBIT CHECK
TEST  7 - REQUEST IN INTERRUPT ROUTINE
TEST  8 - ACTIVE STACK
TEST  9 - REQUEST STACK

END PASS          1
```

```
MODES:     EXTENDED MAPPED
             YES       NO
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
MODES: EXTENDED MAPPED
             YES   YES
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
```

```
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK

END PASS:          10

END CN.INT.r:              10 PASS(ES)
```

```
MODES:     EXTENDED MAPPED
             YES      NO
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
MODES: EXTENDED MAPPED
         YES   YES
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
```

```
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK
TEST 1 - SCRATCHPAD MEMORY CHECKOUT
TEST 2 - BASIC INSTRUCTION CHECKOUT
TEST 3 - INTERRUPT & TRAP CONFIGURATION
TEST 4 - INSTRUCTION CHECKOUT
TEST 5 - LEVEL CONTEXT SWITCHING
TEST 6 - ACTIVE INHIBIT CHECK
TEST 7 - REQUEST IN INTERRUPT ROUTINE
TEST 8 - ACTIVE STACK
TEST 9 - REQUEST STACK

END PASS:          10

END CN.INT.r:              10 PASS(ES)
```

Gould CONCEPT/32™

IPU Trap

(CN.IPT.r)

Diagnostic Description

Diagnostic Number: 111031-242r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005570-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 IPU Trap (CN.IPT.r) Diagnostic Description Release 1.0, Publication Order Number **326-005570-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-4
3-1 through 3-2
4-1 through 4-3/4-4
5-1/5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2
2.3 IPU Execution Modes 2-3
2.4 Memory Configuration 2-3

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test 1--Start IPU (SIP) and IPU Finished (IPF) Traps 4-1
4.3 Test 2--Supervisor Call Trap (SVC) 4-1
4.4 Test 3--Privilege Violation Trap (PRV) 4-1
4.5 Test 4--Map Check Trap (MAP) 4-1
4.6 Test 5--Undefined Instruction Trap (UNI) 4-1
4.7 Test 6--Machine Check Trap (MCK) 4-2
4.8 Test 7--Address Specification Trap (AST) 4-2
4.9 Test 8--Privilege Mode Halt Trap (HLT) 4-2
4.10 Test 9--Arithmetic Exception Trap (ART) 4-2
4.11 Test 10--Nonpresent Memory Trap (NPM) 4-2
4.12 Test 11--Undefined IPU Instruction Trap (UII) 4-2
4.13 Test 12--Alternate Operation 4-2
4.14 Test 13--Parallel (Shared Memory) Operation 4-3
4.15 Test 14--IOP IPU 4-3

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 IPU Trap Diagnostic (CN.IPT.r) is a level one macrodiagnostic program that verifies the context switching mechanism of all software accessible traps. Each test verifies the unique properties of individual traps.

This program operates on the CONCEPT 32/67, 32/87, and 32/97 processors.

Options have been implemented that provide direct operator control over tests to be executed, milestone messages, error looping, and other functions.

Note: This manual describes the specific attributes of Gould CONCEPT/32 IPU Trap Diagnostic (CN.IPT.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU/IPU with a minimum of 64K words of memory.

The program resides at a default starting address of 3000_{16}. This program must not be loaded into any memory module other than zero.

In addition to program storage in main memory, this program requires a 2000_{16} word interrupt dispatch queue.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

The program requires approximately 3 to 7 seconds to execute the first pass and 15 to 30 seconds to execute each subsequent 10 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 IPU Trap Diagnostic (CN.IPT.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console:

```
START CN.IPT.r (#111031-242r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to Chapter 5.

2.2 Option Set

The following option set has been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Unimplemented.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader Start command.
7	Reset	Unimplemented.
8	Reset	Inhibits execution of all tests requiring operator intervention.
9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the operator's console upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Unimplemented.
13	Reset	Unimplemented.
14	Reset	Inhibits interval timer. Activate the interval timer for every pass following pass three by setting this option.
15	Reset	Inhibits all test iterations. Perform a specified number of test iteration on every pass following pass two by resetting this option.
16	Reset	Inhibits all error message printouts.
17	Reset	Inhibits all error report printouts.

Option	Default	Effect When Set
18	Reset	Inhibits automatic exit from Level 9 command file execution if the IPU is not present or is not functioning properly. Inhibiting automatic exit from Level 9 command file execution causes the program to generate an error. To recover from this error enter @@A to recall the Diagnostic Overlay Loader, reload the program, set options 0, 1, and 2, and restart the program. When this option is reset, the program writes a message describing the error to the operator's console and continues level 9 command file processing.
19	Reset	Echoes all messages to the line printer.
20	Reset	Inhbibits the CPU/MM configuration messages.
21	Reset	Writes scratchpad image.
22	Reset	Forces the Interval Timer Clock mode on the first pass of the program. This option overrides option 14 when both are set.
23	Reset	Inhibits the timeout of the IPU by the CPU.
24	Reset	Unimplemented.
25	Reset	Unimplemented.
26	Reset	Causes the program to immediately halt upon error detection to allow for a snapshot of the error location. The program does not write an error message or save the contents of the GPRs. Nullify the halt by entering the Panel mode Run command on the operator's console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27	Reset	Unimplemented.
28	Reset	Executes tests in a random order.
29	Reset	Executes tests in reverse order.
30	Reset	Inhibits repetition of error messages caused by simulated error conditions in iterative test sequences when option 31 is also set. If option 31 is not set, this option is invalid.
31	Reset	Simulates all possible error conditions in the test run by this program. This option verifies and demonstrates test operations and error message routines.

2.3 IPU Execution Modes

This program operates in the following manner:

The first pass runs with the Interval Timer Clock disabled and no iterations. The second pass runs with the Interval Timer Clock enabled and no iterations. The third through the nth pass run with the Interval Timer Clock enabled and each test will iterate the number of times specified by the program.

2.4 Memory Configuration

The program dynamically determines memory size (starting at location zero) and writes the following message to the operator's console:

```
MEMORY CONFIGURATION:
(aaaaaa-aaaaaa) (mmmm.KW)          PRESENT
(aaaaaa-aaaaaa) (mmmm.KW)       NONPRESENT
```

aaaaaa — specifies addresses making up a range in memory that is either present or nonpresent.

mmmm — specifies the amount of memory in decimal kilowords (KW) that makes up an address range.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 IPU Trap Diagnostic (CN.IPT.r), and the default option settings are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error message lists the failing test number, the failing subtest number, the error number, the Program Status Doubleword, the pass count, and the iteration. In addition, the error message lists information specific to the error condition. Section 3.2 contains the standard format for error messages generated during the execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at the location immediately following the location where the error occured. If in subsequent passes the error continues to occur, refer to the source listing to determine the cause of the error.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program are preceded by the following header:

```
**********************************************************************
*TEST xx., SUBTEST xx., ERROR xxxxx., PSD xxxxxxxx,xxxxxxxx
*PASS      xx., ITERATION      xx.
*
```

TEST	indicates the failing test number.
SUBTEST	indicates the failing subtest number.
ERROR	indicates the error number.
PSD	indicates the Program Status Doubleword.
PASS	indicates the pass number where the error occurred.
ITERATION	indicates the iteration where the error occured.

All numbers followed by a period (.) are decimal. Numbers that are not followed by a period are hexadecimal.

Note: Subtest 0 indicates an entire test (no actual subtests).

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould CONCEPT/32 IPU Trap Diagnostic (CN.IPT.r) verifies the context switching of all software accessible traps. Each of the 14 tests verify the unique properties of individual traps. Tests 2, 3, and 8 require operator intervention during the first pass of the program. These tests timeout and the program continues if no response is entered.

4.2 Test 1--Start IPU (SIP) and IPU Finished (IPF) Traps

This test verifies that the CPU receives a Start IPU (SIP) and an IPU Finished (IPF) trap before a timeout occurs to signal the starting and finishing of IPU execution. The test also verifies that the IPU bit in the IPU status word is set during execution.

4.3 Test 2--Supervisor Call Trap (SVC)

This test verifies that an IPU Supervisor Call Trap (SVC) occurs when the IPU executes an SVC 0,0 instruction.

4.4 Test 3--Privilege Violation Trap (PRV)

This test verifies that an IPU Privilege Violation Trap (PRV) occurs when the IPU executes a Halt (privileged) instruction.

4.5 Test 4--Map Check Trap (MAP)

This test verifies that an IPU Map Check Trap (MAP) occurs when the IPU executes a load map (LPDSDCM) where the sequence count of the BPIX and CPIX both equal 0.

4.6 Test 5--Undefined Instruction Trap (UNI)

This test verifies that an IPU Undefined Instruction Trap (UNI) occurs when the IPU executes an undefined instruction (5000).

4.7 Test 6--Machine Check Trap (MCK)

This test verifies that an IPU Machine Check Trap (MCK) occurs when the IPU executes a context change attempting to load the map where the sequence count for the map exceeds maximum map size.

4.8 Test 7--Address Specification Trap (AST)

This test verifies that an IPU Address Specification Trap (AST) occurs when the IPU executes a load doubleword instruction where an odd register number is used (i.e., LD R1,0).

4.9 Test 8--Privilege Mode Halt Trap (HLT)

This test verifies that an IPU Privilege Mode Halt Trap (HLT) occurs when the IPU executes a Halt instruction while privileged and the Halt Trap is enabled.

4.10 Test 9--Arithmetic Exception Trap (ART)

This test verifies that an Arithmetic Exception Trap (ART) occurs when the IPU executes an add bit on R7 when R7=7FFFFFF.

4.11 Test 10--Nonpresent Memory Trap (NPM)

This test verifies that an IPU Nonpresent Memory Trap (NPM) occurs when the IPU executes a load word from nonpresent memory.

4.12 Test 11--Undefined IPU Instruction Trap (UII)

This test verifies that an Undefined IPU Instruction Trap (UII) occurs when the IPU executes the following undefined IPU instructions in the order given:

CD, TD, SIO, TIO, HIO, GRIO, ECWCS, STPIO, RSCHNL, RSCTL, WCWCS, ACI, ECI, DCI, DACI, BEI.

Each instruction is expected to cause an IPU UII trap.

4.13 Test 12--Alternate Operation

This test verifies that the CPU and IPU can alternately execute instructions of the same test routine.

4.14 Test 13—Parallel (Shared Memory) Operation

This test verifies that the CPU and the IPU can concurrently execute two separate test sequences which read, modify, write, and access the same target location (LOC'ACTUAL').

Note: The system must be enabled for shared memory operation in order to run this test successfully.

4.15 Test 14—IOP IPU

This test verifies that the IPU's IOP is functioning properly.

If option 8 (inhibit operator intervention tests) is set, the program skips this test. If option 8 is reset and the test has not been previously run, operator intervention is required to run the test. If option 8 is reset and the test has been successfully completed, the test runs requiring no operator intervention.

Note: This test is valid for the 32/67 processor only.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of Gould CONCEPT/32 IPU Trap Diagnostic (CN.IPT.r).

```
> LOAD CN.IPT.r
* LOADED: 'CN.IPT.r'
  PGM:03000
> STAR CN.IPT.r PASS(1)

  START CN.IPT.r (#111031-242r date)
  (C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
  ALL RIGHTS RESERVED

  CPU TYPE = 32/xx
  IPU TYPE = 32/xx

  OPTIONS REGISTER (LOC 00003044) = 22B00000

  MEMORY CONFIGURATION:
  (000000-1FFFFC) ( 512.KW)   PRESENT
  (200000-FFFFFC) (3584.KW)   NON-PRESENT

    TEST01.      START IPU AND IPU FINISHED TRAPS
    TEST02.      SUPERVISOR CALL TRAP
    TEST03.      PRIVILEGE VIOLATION TRAP
    TEST04.      MAP FAULT TRAP
    TEST05.      UNDEFINED INSTRUCTION TRAP
    TEST06.      MACHINE CHECK TRAP
    TEST07.      ADDRESS SPECIFICATION ERROR TRAP
    TEST08.      PRIVILEGE MODE HALT TRAP
    TEST09.      ARITHMETIC EXCEPTION TRAP
    TEST10.      NONPRESENT MEMORY TRAP
    TEST11.      UNDEFINED IPU INSTRUCTION TRAP
    TEST12.      ALTERNATE OPERATION
    TEST13.      PARALLEL (SHARED MEMORY) OPERATION
    TEST14.      IPU IOP
      (BYPASSED - MANUAL INTERVENTION INHIBITED)

  END PASS:  1

  END CN.IPT.r:          1 PASS(ES)
```

Gould CONCEPT/32

Mapped Memory Management

(CN.MMM.r)

Diagnostic Description

Diagnostic Number: 111031-262r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005580-000

TM CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Mapped Memory Management (CN.MMM.r) Diagnostic Description Release 1.0, Publication Order Number **326-005580-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vi
1-1/1-2
2-1 through 2-4
3-1 through 3-2
4-1 through 4-7/4-8
5-1 through 5-4

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2
2.3 CPU Execution Modes 2-4
2.4 Memory Configuration 2-4
2.5 Scratchpad Configuration 2-4

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Header Information 3-1
3.3 Message Information 3-1
3.4 Report Information 3-2

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test 1--Map Bit Integrity, Zeros 4-1
4.3 Test 2--Map Bit Integrity, Ones 4-1
4.4 Test 3--Map Bit Integrity, Shift Ones 4-1
4.5 Test 4--Map Bit Integrity, Shift Zeros 4-1
4.6 Test 5--LMAP, NOP On Clear PSD Map Bits 4-1
4.7 Test 6--LMAP, Load and Retain Old MIDs 4-2
4.8 Test 7--LMAP, Segment Map Block Count Zero 4-2
4.9 Test 8--LMAP, Ignore MPL Word 0 4-2
4.10 Test 9--LMAP, Do MPL Word 0 (Old), MPL Word 1 4-2
4.11 Test 10--Map (LMAP) The Program 4-2
4.12 Test 11--Map (LPSDCM) The Program 4-2
4.13 Test 12--BPIX Hits, CPIX Hits 4-2
4.14 Test 13--BPIX Hits, CPIX Misses 4-2
4.15 Test 14--BPIX Hits (Increment), CPIX Misses (Decrement) 4-3
4.16 Test 15--Even Lear Hits 4-3
4.17 Test 16--Odd Lear Hits 4-3
4.18 Test 17--Odd Lear Hits, Odd BPIX 4-3
4.19 Test 18--LPSDCM, NOP on Clear PSD Map Bits 4-3
4.20 Test 19--LPSDCM, Load New MIDs 4-4
4.21 Test 20--LPSDCM, Load and Retain Old MIDs 4-4
4.22 Test 21--LPSDCM, Segment Map Block Count Zero 4-4
4.23 Test 22--LPSDCM, Do MPL Word 0 (Old), MPL Word 1 4-4
4.24 Test 23--Access All Available, Contiguous, Mapped Memory Blocks 4-4
4.25 Test 24--Write Protection (Direct Addressing) 4-4
4.26 Test 25--Write Protection (Indirect Addressing) 4-4
4.27 Test 26--No Trap on TMAPR While Mapped 4-4
4.28 Test 27--Limit Register Fault Trap 4-4
4.29 Test 28--Trap on Invalid Map 0 4-5
4.30 Test 29--Nonpresent Memory Trap 4-5
4.31 Test 30--Map Underflow Trap 4-5
4.32 Test 31--Map Overflow Trap 4-5
4.33 Test 32--Map Boundary Valid/Invalid, Valid/Valid 4-5
4.34 Test 33--Logical Address Space Violation 4-5
4.35 Test 34--Instruction/Operand Fetch from NPM 4-5
4.36 Test 35--Halfback 4-6
4.37 Test 36--Trap Context Change 4-6
4.38 Test 37--Trap Context Change MPLO=NPM 4-6
4.39 Test 38--Load Map via Last MPL 4-6
4.40 Test 39--Store and Load File Across Map Boundary 4-6
4.41 Test 40--Load Map via CPIX Only 4-6
4.42 Test 41--Indirect Access of CPIX Mapped Memory 4-6
4.43 Test 42--Base Register Mode Maximum Logical Address 4-6
4.44 Test 43--Manual Effective Address 4-7
4.45 Test 44--Manual Memory Address Virtual (MAV) 4-7
4.46 Test 45--Load Map via Odd/Even Sequence Count 4-7
4.47 Test 46--Load Map via BPIX/CPIX in Nonpresent Memory 4-7
4.48 Test 47--BPIX Hits (Increment), CPIX Misses (Variable) 4-7

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 Mapped Memory Management Diagnostic (CN.MMM.r) is a level one macrodiagnostic program that validates operability and functionality of the Memory Management (MAP) logic.

The program consists of a sequence of tests which check the MAP logic in an order of increasing functional complexity.

Options have been implemented that provide direct operator control over stop on error, loop on error, loop on test, and loop on subtest.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Mapped Memory Management Diagnostic (CN.MMM.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU, an IOP, and a minimum of 64K words of memory.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 4 to 6 minutes to execute each pass depending upon the following:

- machine type
- machine configuration
- option setting

Note: If option 23 (fast mode) is set, the program requires approximately 15 to 20 seconds to execute each pass.

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Mapped Memory Management Diagnostic (CN.MMM.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console and begins execution:

```
START CN.MMM.r (#111031-262r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to Chapter 5.

2.2 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader START command.
7	Reset	Unimplemented.
8	Reset	Inhibits execution of all tests requiring operator intervention.
9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the line printer upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12-13	Reset	Unimplemented.
14	Reset	Inhibits interval timer. Activate the interval timer for every pass following pass three by setting this option.

Option	Default	Effect When Set
15	Reset	Inhibits all iterations. Perform the number iteration specified every pass following pass two by resetting this option.
16	Reset	Inhibits all error message printouts.
17	Reset	Inhibits all error report printouts.
18	Reset	Dumps the map on applicable errors.
19	Reset	Echoes all output to the line printer.
20	Reset	Inhibits the CPU/MM configuration printouts.
21	Reset	Writes scratchpad image.
22	Reset	Forces Interval Timer Clock mode on first pass. (Overrides option 14 if set.)
23	Reset	Runs in a faster, but less complete mode.
24-25	Reset	Unimplemented.
26	Reset	Causes the program to immediately halt once an error has been detected to allow for a snapshot of the error location. The program does not write an error message or save the contents of the GPRs. Nullify the halt by entering the Panel mode Run command on the operator's console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27	Reset	Unimplemented.
28	Reset	Executes tests randomly.
29	Reset	Executes tests in reverse order.
30	Reset	Inhibits repetition of error message caused by simulated error conditions in iterative test sequences if option 31 is set. If option 31 is reset, this option is invalid.
31	Reset	Simulates all possible error conditions that can be generated by this program. This function can be used as a maintenance or debugging aid by verifying and demonstrating test operations and error message routines.

Note: If GPR0 equals any number other than 0 at program start, the program sets the option word to the value contained in GPR0. If GPR0 contains 0, the option word remains unchanged. If GPR0 contains FFFFFFFF, the program restores the default value to the option word.

2.3 CPU Execution Modes

The first pass executes with the Interval Timer Clock disabled and no iterations. The second pass executes with the Interval Timer Clock enabled and no iterations. The third and all subsequent passes execute with the Interval Timer Clock enabled and each test iterating the number of times specified by the individual test within the program.

2.4 Memory Configuration

The memory configuration required by the program may consist of 64K words to 4M words. The program references memory outside of the program area. If option 20 (inhibit CPU/MM configuration printouts) is set (default is reset), the program writes a message to report the memory configuration sensed by the program.

2.5 Scratchpad Configuration

The scratchpad configuration required by the program may contain any combination of device entries and interrupt entries as long as the IOP console and Line Printer controller (if desired) are configured as device addresses $7EFC_{16}$ and $7EF8_{16}$ respectively. If option 21 (write scratchpad image) is set (default is reset), the program writes a message like the following to report the scratchpad configuration sensed by the program.

```
SCRATCHPAD CONFIGURATION:
00(  0) 0E7B0100 0E7B0101 0E7B0102 0E7B0103
04(  4) 0E7A0400 0E7A0401 0E7A0402 0E7A0403
08(  8) 0F790800 0F4E0900 0F4D0A00 0F4C0B00
. .     .      .      .      .
. .     .      .      .      .
. .     .      .      .      .
FC(252) 00000000 00000000 00000000 00FFFFFF
```

Note: Scratchpad location F3 is modified by the program.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Mapped Memory Management (CN.MMM.r), and the default options are set, the program writes out an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error messsage is organized into three areas: an error header, an error message, and in most cases, an error report.

3.2 Header Information

The header consists of the following three lines:

```
**TEST tt, SUBTEST ss, ERROR ee PSD: iiiiiiii-zzzzzzzz(X)
*PASS ppppp, ITERATION iiiii
*OPTIONS REGISTER (LOC 3044) = nnnnnnnn(X)
```

tt	specifies the failing test number.
ss	specifies the failing subtest number.
ee	specifies the error number.
iiiiiiii	specifies the PSD1 at the time the error is detected.
zzzzzzzz	specifies the PSD2 at the time the error is detected.
ppppp	specifies the pass count at the time the error is detected.
iiiii	specifies the iteration count at the time the error is detected.
nnnnnnnn	specifies the setting of the option register at the time the error is detected.
(X)	indicates a hexadecimal number.

3.3 Message Information

The message information is given in a single line as follows:

```
*THIS LINE IS THE ERROR MESSAGE
```

3.4 Report Information

The report consists of the following two lines:

```
*EXP eeeeeeee(X), ACT aaaaaaaa(X), XOR xxxxxxxx(X)
*STOP ON ERROR:  LOC llllllll
```

eeeeeeee	specifies the expected value of the test result.
aaaaaaaa	specifies the actual value of the test result.
xxxxxxxx	specifies the exclusion of expected and actual values.
llllllll	specifies the location at which the program is halted if option 2 is set.
(X)	indicates a hexadecimal number. All locations are given in hexadecimal. All other numbers, those without (X), are decimal.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

The Mapped Memory Management Diagnostic (CN.MMM.r) consists of 47 tests each designed to verify a different memory management function.

- Tests one through 10 and Tests 12 through 17 verify proper functioning of the Load Map (LMAP) instruction.
- Test 11 and Tests 18 through 22 verify proper functioning of the Load Program Status Doubleword and Change Map (LPSDCM) instruction.
- Tests 23 through 47 each verify proper operation of unique map functions.

4.2 Test 1—Map Bit Integrity, Zeros

This test performs a basic data check of the CPU registers. The test loads the registers using a Load Map (LMAP) instruction with a data pattern of all zeros, then checks the registers against the expected value.

4.3 Test 2—Map Bit Integrity, Ones

This test performs a basic data check of the CPU registers. The test loads the registers using an LMAP instruction with a data pattern of all ones, then checks the registers against the expected value.

4.4 Test 3—Map Bit Integrity, Shift Ones

This test performs a basic data check of the CPU registers. The test loads the registers using an LMAP instruction with a 4-bit nibble data pattern. This nibble is shifted through a field of all zeros and the results checked against the expected value.

4.5 Test 4—Map Bit Integrity, Shift Zeros

This test performs a basic data check of the CPU registers. The test loads the registers using an LMAP instruction with a 4-bit nibble data pattern. This nibble is shifted through a field of all ones and the results checked against the expected value.

4.6 Test 5—LMAP, NOP On Clear PSD Map Bits

This test verifies that the LMAP instruction is treated as a NOP (no operation) when attempting to load the map with the Program Status Doubleword (PSD).

4.7 Test 6—LMAP, Load and Retain Old MIDs

This test verifies that the LMAP instruction is treated as a NOP (no operation) when attempting to load the map with the retain bit set.

4.8 Test 7—LMAP, Segment Map Block Count Zero

This test verifies proper operation of the LMAP instruction by attempting to load the map with a sequence count of zero. This should result in the occurrence of a map trap.

4.9 Test 8—LMAP, Ignore MPL Word 0

This test verifies the proper operation of the LMAP instruction by loading the map with the Master Process List (MPL) Word 0 ignored and Bit 0 of MPL Word 1 set. The map should load correctly.

4.10 Test 9—LMAP, Do MPL Word 0 (Old), MPL Word 1

This test verifies the proper operation of the LMAP instruction by loading a partial map while retaining a partial map.

4.11 Test 10—Map (LMAP) The Program

This test verifies the proper context switching operation of the LMAP instruction by executing two Load Map (LMAP) instructions sequentially.

4.12 Test 11—Map (LPSDCM) The Program

This test verifies the proper operation of the Load Program Status Doubleword and Change Map (LPSDCM) instruction with the Program Status Doubleword (PSD) set not to go mapped.

4.13 Test 12—BPIX Hits, CPIX Hits

This test verifies the proper operation of the LMAP instruction. Both available and potential memory are mapped through the BPIX and the CPIX and is checked for hits throughout.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.14 Test 13—BPIX Hits, CPIX Misses

This test verifies the proper operation of the LMAP instruction. Both available and potential memory are mapped through the BPIX and the CPIX. Memory mapped through the BPIX is checked for hits. Memory mapped through the CPIX is checked for misses.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.15 Test 14—BPIX Hits (Increment), CPIX Misses (Decrement)

This test verifies the proper operation of the LMAP instruction. Both available and potential memory are mapped through the BPIX and CPIX. Memory mapped through the BPIX is checked for a one-to-one relationship with the map. Memory mapped through the CPIX is checked for misses and for the map background value. The BPIX is then incremented and the CPIX decremented and checked again.

This test requires several minutes to execute completely. Setting option 23 causes the test to run faster but less completely.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.16 Test 15—Even Lear Hits

This test verifies the proper operation of the LMAP instruction. Both available and potential memory are mapped through the BPIX and the CPIX. All memory is then loaded in a one-to-one mapping relationship in which all CPIX target addresses are even. The map is then checked for hits in the BPIX and even addresses in the CPIX.

This test requires several minutes to execute completely. Setting option 23 causes the test to run faster but less completely.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.17 Test 16—Odd Lear Hits

This test is functionally the same as Test 15 except that all CPIX target addresses are odd.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.18 Test 17—Odd Lear Hits, Odd BPIX

This test verifies the proper operation of the LMAP instruction. Both available and potential memory are mapped through the BPIX and the CPIX. Memory mapped through the BPIX is checked for a one-to-one relationship with the map. Memory mapped through the CPIX is checked with the map background value. The BPIX and the CPIX target addresses are odd and are checked for hits.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.19 Test 18—LPSDCM, NOP on Clear PSD Map Bits

This test verifies that the LPSDCM is treated as a NOP (no operation) instruction when attempting to load the map with a pattern of zeros.

4.20 Test 19—LPSDCM, Load New MIDs

This test verifies proper operation of the load map instruction LPSDCM by successfully loading the map.

4.21 Test 20—LPSDCM, Load and Retain Old MIDs

This test verifies that the load map instruction LPSDCM is treated as a NOP (no operation) when attempting to load the map with the retain bit set.

4.22 Test 21—LPSDCM, Segment Map Block Count Zero

This test verifies that when attempting to use the LPSDCM instruction to load the map with a sequence count of zero, a map trap occurs.

4.23 Test 22—LPSDCM, Do MPL Word 0 (Old), MPL Word 1

This test verifies proper operation of the load map instruction LPSDCM by loading a partial map while retaining a partial map.

4.24 Test 23—Access All Available, Contiguous, Mapped Memory Blocks

This test verifies that all mapped memory blocks in present memory can be accessed.

4.25 Test 24—Write Protection (Direct Addressing)

This test consists of eight subtests which verify proper operation of the write protection function through direct addressing.

4.26 Test 25—Write Protection (Indirect Addressing)

This test consists of eight subtests which verify proper operation of the write protection function through indirect addressing.

4.27 Test 26—No Trap on TMAPR While Mapped

This test verifies that the TMAPR instruction does not cause a map trap when executed while in the mapped mode.

4.28 Test 27—Limit Register Fault Trap

This test consists of four subtests that verify that attempting to address unmapped memory causes a limit register fault map trap.

4.29 Test 28--Trap on Invalid Map 0

This test verifies that attempting to address an invalid map causes a map trap to occur.

4.30 Test 29--Nonpresent Memory Trap

This test verifies that attempting an LPSDCM when MPL Word 0 is pointing to nonpresent memory causes a nonpresent memory trap to occur.

4.31 Test 30--Map Underflow Trap

This test verifies that attempting to load the map if the sum of the BPIX and CPIX sequence counts is zero causes an underflow map trap to occur.

4.32 Test 31--Map Overflow Trap

This test verifies that attempting to load the map if the sum of the BPIX and CPIX sequence counts exceeds the size of the map causes an overflow map trap to occur.

4.33 Test 32--Map Boundary Valid/Invalid, Valid/Valid

This test consists of 38 subtests.

- Subtests 1 through 19 verify that a Demand Page Fault Trap occurs when executing instructions in an invalid map block. The subtests map two 2KW memory blocks, one valid and the other invalid.
- Subtests 20 through 38 verify that the map register and map image descriptor are updated and expected values compared through the CPU miss logic.

4.34 Test 33--Logical Address Space Violation

This test consists of two subtests which verify that a map trap occurs when logical address space is violated. The test maps two 2KW memory blocks, one just below 128KW and one just above. The block above the 128KW boundary is out of the range of logical address and should cause a map trap to occur.

4.35 Test 34--Instruction/Operand Fetch from NPM

This test consists of eleven subtests which verify that a nonpresent memory (NPM) trap occurs when map block boundaries are violated. The test maps two 2KW memory blocks, one just below 128KW and one in the first nonpresent memory block. The block within nonpresent memory creates a map block boundary violation and should cause a NPM trap to occur.

This test is skipped if nonpresent memory does not exist.

4.36 Test 35--Halfback

This test verifies that a series of contiguous halfword instructions execute in all mapped memory blocks.

4.37 Test 36--Trap Context Change

This test verifies context changing into and out of mapped mode.

4.38 Test 37--Trap Context Change MPL0=NPM

This test verifies that attempting to context change into mapped mode with MPL word 0 pointing to nonpresent memory causes a machine check trap to occur.

4.39 Test 38--Load Map via Last MPL

This test verifies that mapping via the last available MPL word is successful.

4.40 Test 39--Store and Load File Across Map Boundary

This test verifies that a store and load file can be successfully executed across a map boundary.

4.41 Test 40--Load Map via CPIX Only

This test verifies the proper operation of the LPSDCM instruction. The map is loaded using only the CPIX. No traps should occur.

4.42 Test 41--Indirect Access of CPIX Mapped Memory

This test consists of five subtests that verify the ability to indirectly access memory mapped through the CPIX. The test maps both available and potential memory through the BPIX and the CPIX. The results of stores, fetches, and branches of the CPIX are then checked to show that they are correct and that no traps occur.

4.43 Test 42--Base Register Mode Maximum Logical Address

This test consists of two subtests that verify the proper operation of the Base Register mode in relation to the maximum mapped logical address. The test loads all base registers and starts execution Mapped, in Base Register mode. Upon completion, Mapped Base Register mode is disabled, base registers are stored and verified, and execution results are checked.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.44 Test 43—Manual Effective Address

This test verifies the manual Effective Address capabilities. The test writes the Effective Address (EA) command and the Effective Address to the operator's console. The CPU performs a compare and verifies proper operation of the manual Effective Address capabilities.

Setting option 8 (inhibit operator intervention tests) causes the program to skip this test.

Note: This test is valid for the CONCEPT 32/67 and CONCEPT 32/97 only.

4.45 Test 44—Manual Memory Address Virtual (MAV)

This test verifies the ability to manually address CONCEPT/32. The Memory Address Virtual (MAV) command and the CONCEPT/32 Address are written to the operator's console.

Setting option 8 (inhibit operator intervention tests) causes the program to skip this test.

4.46 Test 45—Load Map via Odd/Even Sequence Count

This test consists of two subtests which verify that a Load Map (LMAP) instruction is executed with an odd and even BPIX and CPIX count. Subtest 1 loads the map in an odd BPIX/CPIX sequence count. Subtest 2 loads the map in an even BPIX/CPIX sequence count.

4.47 Test 46—Load Map via BPIX/CPIX in Nonpresent Memory

This test consists of four subtests. Subtest 1 verifies that a Map Trap occurs when the BPIX and CPIX are residing in nonpresent memory. Subtests 2, 3, and 4 verify the inablility to execute fetches, stores, and branches of the CPIX which is pointing to nonpresent memory.

4.48 Test 47—BPIX Hits (Increment), CPIX Misses (Variable)

This test verifies the proper operation of the LMAP instruction. Both available and potential memory are mapped through the BPIX and CPIX. Memory mapped through the BPIX is checked for a one-to-one relationship with the map. Memory mapped through the CPIX is checked for misses and for the map background valve. The BPIX is then incremented and the variable assigned to the CPIX is checked again.

This test requires several minutes to execute completely. Setting option 23 causes the test to run faster but less completely.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of the Gould CONCEPT/32 Mapped Memory Management Diagnostic (CN.MMM.r).

```
>LOAD CN.MMM.r
*LOADED: 'CN.MMM.r'
 PGM: 03000
>STAR CN.MMM.r PASS(1)

START CN.MMM.r (#111031-262r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

OPTIONS REGISTER (LOC 00003044) = 22B00100

MEMORY CONFIGURATION:
(000000-1FFFFC) ( 512.KW)     PRESENT
(200000-FFFFFC) (3584.KW)  NON-PRESENT

    TEST01. MAP BITS INTEGRITY, ZEROS
    TEST02. MAP BITS INTEGRITY, ONES
    TEST03. MAP BITS INTEGRITY, SHIFT ONES
     SUBTEST: 1.
     SUBTEST: 2.
     SUBTEST: 3.
     SUBTEST: 4.
    TEST04. MAP BITS INTEGRITY, SHIFT ZEROS
     SUBTEST: 1.
     SUBTEST: 2.
     SUBTEST: 3.
     SUBTEST: 4.
    TEST05. LMAP: NOP ON CLEAR PSD MAP BITS
    TEST06. LMAP: LOAD AND RETAIN OLD MIDS
    TEST07. LMAP: SEGMENT MAP BLOCK COUNT ZERO
    TEST08. LMAP: IGNORE MPL WORD 0
    TEST09. LMAP: DO MPL WD 0 (OLD), MPL WD 1
    TEST10. MAP (LMAP) THE PROGRAM
    TEST11. MAP (LPSDCM) THE PROGRAM
    TEST12. BPIX HITS, CPIX HITS
    TEST13. BPIX HITS, CPIX MISSES
    TEST14. BPIX HITS (INCR), CPIX MISSES (DECR)
```

```
TEST15. EVEN LEAR HITS
TEST16. ODD LEAR HITS
TEST17. ODD LEAR HITS, ODD BPIX
TEST18. LPSDCM: NOP ON CLEAR PSD MAP BITS
TEST19. LPSDCM: LOAD NEW MIDS
TEST20. LPSDCM: LOAD AND RETAIN OLD MIDS
TEST21. LPSDCM: SEGMENT MAP BLOCK COUNT ZERO
TEST22. LPSDCM: DO MPL WD 0 (OLD), MPL WD 1
TEST23. ACCESS ALL AVAIL CONTIG MAPPED MEM BLOCKS
TEST24. WRITE PROTECTION
 SUBTEST: 1.
 SUBTEST: 2.
 SUBTEST: 3.
 SUBTEST: 4.
 SUBTEST: 5.
TEST25. WRITE PROTECTION, INDIRECT
 SUBTEST: 1.
 SUBTEST: 2.
 SUBTEST: 3.
 SUBTEST: 4.
 SUBTEST: 5.
TEST26. TRAP ON TMAPR WHILE MAPPED
TEST27. LIMIT REGISTER FAULT TRAP
 SUBTEST: 1.
 SUBTEST: 2.
 SUBTEST: 3.
 SUBTEST: 4.
TEST28. TRAP ON INVALID MAP 0
TEST29. NOT PRESENT MEMORY TRAP
 (BYPASSED - N/A WHEN NPM =  80000)
TEST30. MAP UNDERFLOW TRAP
TEST31. MAP OVERFLOW TRAP
TEST32. MAP BOUNDARY VALID/INVALID, VALID/VALID
 SUBTEST:  1.
 SUBTEST:  2.
 SUBTEST:  3.
 SUBTEST:  4.
 SUBTEST:  5.
 SUBTEST:  6.
 SUBTEST:  7.
 SUBTEST:  8.
 SUBTEST:  9.
 SUBTEST: 10.
 SUBTEST: 11.
 SUBTEST: 12.
 SUBTEST: 13.
 SUBTEST: 14.
 SUBTEST: 15.
```

```
 SUBTEST: 16.
 SUBTEST: 17.
 SUBTEST: 18.
 SUBTEST: 19.
 SUBTEST: 20.
 SUBTEST: 21.
 SUBTEST: 22.
 SUBTEST: 23.
 SUBTEST: 24.
 SUBTEST: 25.
 SUBTEST: 26.
 SUBTEST: 27.
 SUBTEST: 28.
 SUBTEST: 29.
 SUBTEST: 30.
 SUBTEST: 31.
 SUBTEST: 32.
 SUBTEST: 33.
 SUBTEST: 34.
 SUBTEST: 35.
 SUBTEST: 36.
 SUBTEST: 37.
 SUBTEST: 38.
TEST33. LOGICAL ADDRESS SPACE VIOLATION
 SUBTEST:  1.
 SUBTEST:  2.
TEST34. INSTRUCTION/OPERAND FETCH FROM NPM
 SUBTEST:  1
 SUBTEST:  2
 SUBTEST:  3
 SUBTEST:  4
 SUBTEST:  5
 SUBTEST:  6
 SUBTEST:  7
 SUBTEST:  8
 SUBTEST:  9
 SUBTEST: 10
 SUBTEST: 11
TEST35. HALFBACK
TEST36. TRAP CONTEXT CHANGE
TEST37. TRAP CONTEXT CHANGE (MPL0=NPM)
TEST38. LOAD MAP VIA LAST MPL
TEST39. STORE AND LOAD FILE ACROSS MAP BOUNDARY
TEST40. LOAD MAP VIA CPIX ONLY
TEST41. INDIRECT ACCESS OF CPIX MAPPED MEMORY
 SUBTEST:  1.
 SUBTEST:  2.
 SUBTEST:  3.
 SUBTEST:  4.
 SUBTEST:  5.
TEST42. BR MODE MAXIMUM LOGICAL ADDRESS
```

```
TEST43. MANUAL EFFECTIVE ADDRESS (EA)
 (BYPASSED - MANUAL INTERVENTION INHIBITED)
TEST44. MANUAL MEMORY ADDRESS VIRTUAL (MAV)
 (BYPASSED - MANUAL INTERVENTION INHIBITED)
TEST45. LOAD MAP VIA ODD/EVEN SEQUENCE COUNT
 SUBTEST: 1.
 SUBTEST: 2.
TEST46. LOAD MAP VIA BPIX/CPIX IN NONPRESENT MEMORY
 SUBTEST: 1.
 SUBTEST: 2.
 SUBTEST: 3.
 SUBTEST: 4.



        TEST47. BPIX HITS (INCR), CPIX MISSES (VARI)
END PASS     1.  - FAST MODE -
  (ERRORS:     0. SINCE LAST REPORT,     0. SINCE PROG START)


*****INTERVAL TIMER MODE IS ENABLED*****

END CN.MMM.r:          1. PASS(ES)
```

Gould CONCEPT/32™

Multiprocessor Shared Memory

(CN.MSM.r)

Diagnostic Description

Diagnostic Number: 111031-253r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005600-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Multiprocessor Shared Memory (CN.MSM.r) Diagnostic Description Release 1.0, Publication Order Number **326-005600-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vii/viii
1-1/1-2
2-1 through 2-8
3-1/3-2
4-1 through 4-12
5-1 through 5-2

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Program Initialization 2-2
2.3 Loading the Program into a Node 2-5
2.4 Option Set 2-6
2.5 Single CPU Configuration Execution 2-8
2.6 Multiple CPU Configuration Execution 2-8

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Description of Error Message 3-1

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Testing Modes 4-1
4.3 Data Pattern Type Tests 4-2
4.3.1 Test 1--Store Data Pattern Using All Zeros 4-2
4.3.2 Test 2--Store Data Pattern Using All Ones 4-2
4.3.3 Test 3--Store Data Pattern Using Alternating Ones (X'55555555') 4-2
4.3.4 Test 4--Store Data Pattern Using Alternating Ones (X'AAAAAAAA') 4-2
4.3.5 Test 5--Store Data Pattern Using Address in Address 4-2
4.3.6 Test 6--Store Data Pattern for Refresh Using All Ones 4-2
4.3.7 Test 7--Store Data Pattern for Refresh Using All Zeros 4-3
4.3.8 Test 8--Store Data Pattern For Refresh Using Alternating Ones (X'55555555') 4-3
4.3.9 Test 9--Store Data Pattern For Refresh Using Alternating Ones (X'AAAAAAAA') 4-3
4.3.10 Test 10--Store Data Pattern For Refresh Using Address in Address 4-3
4.3.11 Test 11--Moving Inversions With Seed X'11111111' 4-3
4.3.12 Test 12--Moving Inversions With Seed X'AAAAAAAA' 4-4
4.3.13 Test 13--Moving Inversions With Seed X'55555555' 4-4
4.3.14 Test 14--Moving Inversions With Seed X'AAAAAAAA' 4-4
4.3.15 Test 15--Moving Inversions With Seed X'AAAAAAAA' 4-4
4.3.16 Test 16--F and C Bit Test STB/LW 4-5
4.3.17 Test 17--F and C Bit Test STH/LW 4-5
4.3.18 Test 18--F and C Bit Test STW/LB 4-5
4.3.19 Test 19--F and C Bit Test STW/LH 4-5
4.3.20 Test 20--F and C Bit Test STD/LW 4-6
4.3.21 Test 21--F and C Bit Test STW/LD 4-6
4.3.22 Test 22--Error Correction Code Check Using ECC Bit 14 4-6
4.3.23 Test 23--Error Correction Code Check Using ECC Bit 18 4-6
4.3.24 Test 24--Error Correction Code Check Using ECC Bit 19 4-6

4.4 Multiprocessor Type Tests 4-7
4.4.1 Test 1--Memory Execution Test (Instruction Fetch-Path) 4-7
4.4.2 Test 2--Cache Write-thru Capability - Primary On 4-7
4.4.3 Test 3--Cache Write-thru Capability - Primary Off 4-7
4.4.4 Test 4--Cache Write-thru Capability - Seconday On 4-7
4.4.5 Test 5--Cache Write-thru Capability - Secondary Off 4-8
4.4.6 Test 6--Memory Execution Test with Contention (Instruction Fetch-Path) - Secondary 4-8
4.4.7 Test 7--Memory Execution Test with Contention (Instruction Fetch-Path) - Primary 4-8
4.4.8 Test 8--Memory Contention Test (Instruction Fetch-Path) - Secondary 4-8
4.4.9 Test 9--Memory Contention Test (Instruction Fetch-Path) - Primary 4-9
4.4.10 Test 10--Memory Contention Test (Zero/Set Bit in Memory) with Iteration 4-9
4.4.11 Test 11--Memory Contention Test (Set Bit in Memory) - Primary Cache On 4-10
4.4.12 Test 12--Critical Section Test - Cache Off 4-10
4.4.13 Test 13--Critical Section Test - Cache On 4-10
4.5 Manual Intervention Tests 4-11
4.5.1 Manual Intervention Test 1-Battery Back-up 4-11
4.5.2 Manual Intervention Test 2-ECC Switches 4-12

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 Multiprocessor Shared Memory Diagnostic (CN.MSM.r) is a level one macrodiagnostic program that verifies the proper operation of the Multiprocessor Shared Memory system. The program contains a sequence of tests which verify the multiprocessor's memory ports, controllers, and storage boards.

This program performs memory testing using mapped addressing. Nonpresent memory is ignored and can be jumpered within testing ranges.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Multiprocessor Shared Memory Diagnostic (CN.MSM.r). For a discussion of the level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires at least one CONCEPT/32 CPU with at least 64K words of memory to test private memory configurations. For shared memory configurations, this program requires the following:

- one model 3050 or 3051 Multiprocessor Shared Memory System
- two CONCEPT/32 CPUs with at least 64K words of private memory
- one model 3055 or 3056 Multiprocessor Shared Memory Port kit for each CPU
- One model 3060 or 3061 Memory Module

1.3 Execution Time

This program requires approximately 30 to 45 minutes to execute depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould Multiprocessor Shared Memory Diagnostic (CN.MSM.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following overview lists the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the the initial program load (IPL) sequence and the load and start procedures, the program writes the following messages to the operator's console and begins its initialization sequence.

```
START CN.MSM.r (#111031-253r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION.
ALL RIGHTS RESERVED
CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

```
WARNING!  INSURE FOLLOWING SWITCHES ARE ENGAGED (SWITCHED TO LEFT)
          ON SHARED MEMORY SYSTEM :

          ALL ERROR          (REPORT ALL ERRORS).
          DISECC             (DISABLE ECC LOGIC).
          OPTIONS REGISTERS (LOC xxxxxxxx) = optword
          TEST EXECUTION CONTROL LIST (LOC xxxxxxxx) = teclword
          MEMORY CONFIGURATION:
          (frstaddr - lastaddr) (size.KW) PRESENT
          (frstaddr - lastaddr) (size.KW) NONPRESENT
```

xxxxxxxx specifies the memory location at which the word (either option word or test execution control list doubleword) resides.

optword specifies the TCA option word.

teclword specifies the test execution control list (TECL) doubleword.

frstaddr specifies the first memory address of the memory range (either present or nonpresent).

lastaddr specifies the last memory address of the memory range (either present or nonpresent).

size specifies the amount of memory, in K words, available for the memory range (either present or nonpresent).

2.2 Program Initialization

Following the program start and CPU type messages, program initialization begins.

1. The program issues the following prompt to the operator's console:

   IS THIS DIAGNOSTIC IMAGE LOADED IN A PRIMARY CPU OR A
   SECONDARY CPU (NODE) ?
   ENTER "P" FOR PRIMARY CPU OR
   JUST CR FOR DEFAULT OF SECONDARY NODE

   Enter the correct response for the CPU type under test.

   If the CPU type is a primary processor, the initialization sequence continues with steps two through five, otherwise, it jumps to step six. The program repeats steps six through 13 for each secondary processor configured. The program writes the secondary CPU initialization prompts to the primary processor's console for each secondary processor (node) tested.

2. The program issues the following prompt to the operator's console:

   PRIMARY PROCESSOR'S PRIVATE MEMORY START ADDRESS ?
   ENTER 0 FOR NO PRIVATE MEMORY TESTING THEN CR OR
   ENTER HEX. WORD ADDRESS THEN CR OR
   JUST CR FOR DEFAULT OF xxxxxxxx>

   xxxxxxxx specifies the hexadecimal address of the first word location, after 1K word increment, from end of program.

3. The program issues the following prompt to the operator's console:

   PRIMARY PROCESSOR'S PRIVATE MEMORY STOP ADDRESS ?
   ENTER HEX. WORD ADDRESS THEN CR OR
   JUST CR FOR DEFAULT OF xxxxxxxx>

   xxxxxxxx specifies the hexadecimal address of the next valid map block or 1K word increment from the primary processor's private memory start address.

4. The program issues the following prompt to the operator's console:

   PRIMARY PROCESSOR'S SHARED MEMORY START ADDRESS ?
   ENTER 0 FOR NO SHARED MEMORY TESTING THEN CR OR
   ENTER HEX. WORD ADDRESS THEN CR OR
   JUST CR FOR DEFAULT OF xxxxxxxx>

   xxxxxxxx specifies the hexadecimal address of the first location after end of the primary processor's private memory.

5. The program issues the following prompt to the operator's console:

   PRIMARY PROCESSOR'S SHARED MEMORY STOP ADDRESS ?
   ENTER HEX. WORD ADDRESS THEN CR OR
   JUST CR FOR DEFAULT OF xxxxxxxx>

   xxxxxxxx specifies the hexadecimal address of the first doubleword location after 8K word increment from the primary processor's shared memory start address.

6. The program issues the following prompt to the operator's console:

   HOW MANY SECONDARY CPUS (NODES) ARE CONFIGURED ?
   ENTER DECIMAL NUMBER INCLUSIVELY BETWEEN 0 AND 15 THEN CR OR
   JUST CR FOR DEFAULT OF 0>.

The primary processor executes memory verification type tests on it's private and shared memory.

If secondary processors are configured, the following messages are written to primary processor's console for each secondary processor (node) configured.

7. The program issues the following prompt to the operator's console:

   SECONDARY CPUnn (NODE) PRIVATE START ADDRESS ?
   ENTER 0 FOR NO PRIVATE MEMORY TESTING THEN CR OR
   ENTER HEX. WORD ADDRESS THEN CR OR
   JUST CR FOR DEFAULT OF xxxxxxx>

nn — specifies the program assigned secondary processor designator identifying each secondary processor.

xxxxxxxx — specifies the precalculated hexadecimal address of the first word location, after 1K word increment, from the end of program.

8. The program issues the following prompt to the operator's console:

   SECONDARY CPUnn (NODE) PRIVATE STOP ADDRESS ?
   ENTER HEX. WORD ADDRESS THEN CR OR
   JUST CR FOR DEFAULT OF xxxxxxxx>

nn — specifies the program assigned secondary processor designator identifying each secondary processor.

xxxxxxxx — specifies the precalculated hexadecimal address of the next valid map block of 1K word increment, from the primary processor's private memory start address.

9. The program issues the following prompt to the operator's console:

   SECONDARY CPUnn (NODE) SHARED MEMORY START ADDRESS ?
   ENTER 0 FOR NO SHARED MEMORY TESTING THEN CR OR
   ENTER HEX. WORD ADDRESS THEN CR OR
   JUST CR FOR DEFAULT OF primaddr>

primaddr — specifies the address assigned to the primary processor.

10. The program issues the following prompt to the operator's console:

    SECONDARY CPUnn (NODE) SHARED MEMORY STOP ADDRESS ?
    ENTER HEX. WORD ADDRESS THEN CR OR
    JUST CR FOR DEFAULT OF primaddr>

primaddr — specifies the address assigned to the primary processor.

When secondary processors exist, the program continues with secondary processor configuration provided no operator intervention response errors are encountered.

11. The program issues the following prompt to the operator's console:

    SECONDARY CPUnn (NODE) TO BE LOADED FROM DEVICE OR
    COPIED FROM SHARED MEMORY VIA "THUMB-IN"? ENTER "T" FOR
    THUMB-IN THEN CR OR JUST CR FOR DEFAULT OF LOAD FORM DEVICE.>

nn — specifies the program assigned secondary processor designator identifying each secondary processor.

When loading from a device, perform an initial program load (IPL) from floppy disc or tape. If no floppy disc or magnetic tape is configured, refer to Section 2.3 for thumb-in procedure. Otherwise, the initialization sequence continues.

12. If one or more secondary processor is configured, the program issues the following prompt to the operator's console:

    SECONDARY CPUnn IS ASSIGNED THE COMMUNICATION BUFFER
    STARTING AT ADDRESS xxxxxxxx>

xxxxxxxx — specifies the primary processor assigned, precalculated, hexadecimal address of the secondary processor's communication buffer.

The program writes the following message to the secondary processor's operator's console:

    COMMUNICATION BUFFER STARTING ADDRESS?
    ENTER ON THIS CONSOLE BUFFER ADDRESS GIVEN ON PRIMARY
    PROCESSOR'S CONSOLE.>

Enter the address shown on the primary processor's operator's console.

During each secondary processor's initialization sequence, the program performs communication checks to ensure each secondary processor can communicate with the primary processor without the primary timing-out. These checks consist of signal and wait algorithms. The program executes this combination of communication checks one hundred times for each secondary processor.

When a communication link is established, the program writes the following message to both the primary and secondary processor's operator's console:

    SECONDARY CPUnn (NODE) COMMUNICATION LINK CONFIRMED.

nn — specifies the program assigned secondary processor designator identifying each secondary processor.

Note: The secondary processors's communication area is the first 32-word block of its shared memory. Each additional secondary configured with common shared memory assumes a communication area of a 32-word block after previous communication area entry.

13. The program issues the following prompt to the operator's console:

    IS CACHE WRITE-THRU ENABLED ON THIS
    MULTIPROCESSOR SHARED MEMORY SYSTEM?
    ENTER "N" FOR NO OR JUST CR FOR DEFAULT OF YES.>

2.3 Loading the Program into a Node

A thumb-in procedure provides the ability to load the secondary processor from shared memory when no floppy disc or magnetic tape is configured. The sequence for loading via the thumb-in procedure follows:

The program writes the following message to the primary processor's operators console:

ENTER THE FOLLOWING "THUMB-IN' STARTING AT ADDRESS xxxxxxxx in CPUnn's PRIVATE MEMORY

ADDRESS	INSTRUCTION
xxxxxxxx	000D0002
xxxxxxxx	AEC00000
xxxxxxxx	D6Eyyyyy
xxxxxxxx	C9010004
xxxxxxxx	F5Cyyyyy
xxxxxxxx	ECyyyyyy

xxxxxxxx — specifies the primary processor assigned, precalculated, hexadecimal address of the secondary processor's private memory thumb-in starting location.

yyyyy — specifies the primary processor assigned, precalculated, hexadecimal address of the secondary processor's private memory thumb-in starting location.

Enter the address specified by xxxxxxxx, on the secondary processor's operator's console. The program issues the following message to the primary processor's operator's console:

DO A SYSTEM RESET (RST) THEN
ENTER GPR2=xxxxxxGPR3=yyyyyyyyPC=zzzzzz
THEN ENTER RUN.

xxxxxx — specifies the contents of general purpose register (GPR) 2.

yyyyyyyy — specifies the contents of general purpose register (GPR) 3.

zzzzzz — specifies the program counter assignment.

The program issues the following message to the operator's console:

ENTER CR WHEN DIAGNOSTIC IS LOADED.

2.4 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Unimplemented
5	Reset	Unimplemented
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader Start command.
7	Reset	Unimplemented.
8	Reset	Inhibits execution of all tests requiring operator intervention.
9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the line printer upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12-14	Reset	Unimplemented.

Option	Default	Effect When Set
15	Reset	Inhibits all iterations.
16	Reset	Inhibits all error message printouts.
17	Reset	Inhibits all error report printouts.
18	Reset	Unimplemented.
19	Reset	Echoes all messages to the line printer.
20	Reset	Verifies the addressability, data retention and execution of memory. The program verifies every memory location, except those in which the Diagnostic Overlay Loader (DOL) resides (including all designated test areas and the program's point of initial residence) by relocating the program through the primary processor's private and shared memory. The program executes data pattern type tests in each memory location and continues to relocate itself until this option is reset.
21	Reset	If option 20 is set, this option allows only one complete relocation of the program. Once the program relocates itself through every memory location in the address range being tested, program execution continues from its point of initial residence.
22	Reset	Causes the program to use variable TECL doubleword dependent memory size and count.
23-25	Reset	Unimplemented.
26	Reset	Causes the program to immediately halt upon error detection to allow for a snapshot of the error location. The program does not write an error message or save the contents of the GPR's. Nullify the halt by entering the Panel mode Run command on the operator's console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27	Reset	Unimplemented.
28	Reset	Executes tests in a random order.
29	Reset	Executes tests in reverse order.
30-31	Reset	Reserved - Do not use.

2.5 Single CPU Configuration Execution

No special considerations are required for single CPU execution (private memory testing).

2.6 Multiple CPU Configuration Execution

In multiple CPU configurations, the primary processor controls the execution of the secondary CPUs by manipulating various control flags and monitoring status flags. The primary processor is also responsible for writing background data patterns for copying blocks of test code to shared memory for testing by the secondary processors, and for reporting all events experienced by the secondary processors.

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Multiprocessor Shared Memory (CN.MSM.r), and the default options are set, the program writes out an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

The error messages lists the failing test number, the failing subtest number, the error number, the Program Status Doubleword, the pass count, and the iteration. In addition, the error message lists information specific to the error condition. Section 3.2 contains the standard format for error messages generated during the execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at the location immediately following the location where the error occured. If in subsequent passes the error continues to occur, refer to the source listing to determine the cause of the error.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors generated during the execution of this program are preceded by the following header:

```
**********************************************************************
*TEST xx.,     SUBTEST xx.,     ERROR xxxxx.,     PSD xxxxxxxx,xxxxxxxx
*PASS xx.     ITERATION xx.
```

TEST	indicates the failing test number.
SUBTEST	indicates the failing subtest number.
ERROR	indicates the error number.
PSD	indicates the Program Status Doubleword.
PASS	indicates the pass number where the error occurred.
ITERATION	indicates the iteration where the error occurred.

All numbers followed by a period are decimal. Numbers that are not followed by a period are hexadecimal numbers.

Error messages which follow the above header relate the specific error condition. If more than one error is generated, the program writes out the above header for error message generated.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

The Gould CONCEPT/32 Multiprocessor Shared Memory Diagnostic (CN.MSM.r) verifies the ability of the Multiprocessor Memory ports, controllers, and storage boards to communicate data between the SelBUS and remote memory. This program also verifies cache the write-through ability and error correction abilities of the Multiprocesser Memory ports, controllers and storage boards.

The Test Execution Control List (TECL) runs and controls all tests. Upon completion of the initial program load (IPL), the TECL defaults to all scheduled tests. To modify the TECL, enter @@P on the operator's console, modify the TECL word, and enter Reset (cr) and Run (cr) on the operator's console.

Bits 0 through 23 of the TECL doubleword correspond to data pattern type tests 1 through 24, respectively (refer to Section 4.3). Bits 24 through 36 correspond to the multiprocessor type tests 1 through 13, respectively (refer to Section 4.4). Setting a bit in the TECL doubleword inhibits the test to which the bit corresponds from executing. Inhibit manual intervention tests by setting option 8 (inhibit operator intervention tests). The following is an example of a TECL doubleword.

```
TCA.TECL   DATAD    FE0FFF00F8
```

In this example, only data pattern type tests 8 through 12 (corresponding to bits 7 through 11) execute. Multiprocessor type tests 1 through 8 (corresponding to bits 24 through 31) execute. Bits 37 through 39 are undefined.

4.2 Testing Modes

Test execution is organized into two execution modes -- data pattern type tests and multiprocessor type tests. The primary processor is responsible for scheduling the mode to be executed by all secondary processors.

If no secondary processors are configured, the primary processor executes only data pattern type tests.

If one or more secondary processors are configured, the program executes data pattern type tests in a group. If option 22 is set, the size of this group of tests is dependent upon the size of memory and the number of secondary processors configured. This group of tests executes once for each secondary processor and is scheduled sequentially by the primary processor.

Following execution of the data pattern type tests by each secondary processor, all processors enter the Multiprocessor Type Testing mode (refer to Section 4.4).

4.3 Data Pattern Type Tests

If no secondary processors are configured, the program executes data pattern type tests once on the primary processor.

If a secondary processor is configured, the program executes data pattern type tests once on the primary processor then once, in a group, for each secondary processor.

4.3.1 Test 1--Store Data Pattern Using All Zeros

This test uses a simple data pattern write/read to every location between the start and stop addresses using the following data pattern:

X'00000000'

4.3.2 Test 2--Store Data Pattern Using All Ones

This test uses a simple data pattern write/read to every location between the start and stop addresses using the following data pattern:

X'FFFFFFFF'

4.3.3 Test 3--Store Data Pattern Using Alternating Ones (X'55555555')

This test uses a simple data pattern write/read to every location between the start and stop addresses using the following data pattern:

X'55555555'

4.3.4 Test 4--Store Data Pattern Using Alternating Ones (X'AAAAAAAA')

This test uses a simple data pattern write/read to every location between the start and stop addresses using the following data pattern:

X'AAAAAAAA'

4.3.5 Test 5--Store Data Pattern Using Address In Address

This test uses a simple write/read address-in-address over the entire valid address-space.

4.3.6 Test 6--Store Data Pattern For Refresh Using All Ones

This test verifies integrity of memory over a period of time by performing a write/pause/read to every location between the start and stop addresses with the following data pattern:

X'FFFFFFFF'

4.3.7 Test 7—Store Data Pattern For Refresh Using All Zeros

This test verifies integrity of memory over a period of time by performing a write/pause/read to every location between the start and stop addresses with the following data pattern:

X'00000000'

4.3.8 Test 8—Store Data Pattern For Refresh Using Alternating Ones (X'55555555')

This test verifies integrity of memory over a period of time by performing a write/pause/read to every location between the start and stop addresses with the following data pattern:

X'55555555'

4.3.9 Test 9—Store Data Pattern For Refresh Using Alternating Ones (X'AAAAAAAA')

This test verifies integrity of memory over a period of time by performing a write/pause/read to every location between the start and stop addresses with the following data pattern:

X'AAAAAAAA'

4.3.10 Test 10—Store Data Pattern For Refresh Using Address In Address

This test verifies integrity of memory over a period of time by performing a write/pause/read to every location between the start and stop addresses with word-address in address.

4.3.11 Test 11—Moving Inversions With Seed X'11111111'

This test executes a read/write/read to every location between the start and stop addresses. Addressing is executed in both the forward and reverse directions. The starting "seed" data pattern is X'11111111'.

The memory is first loaded with zeros from the start and stop addresses. A data pattern is loaded and verified from the start to stop addresses (forward addressing). Another data pattern is then loaded and verified from the stop to start addresses (reverse addressing).

4.3.12 Test 12--Moving Inversions With Seed X'AAAAAAAA'

This test executes a read/write/read to every location between the start and stop addresses. Addressing is executed in both the forward and reverse directions. The starting "seed" data pattern is X'AAAAAAAA'.

The test loads memory with zeros from the start and stop addresses, first loading and verifying a data pattern from the start to stop addresses (forward addressing), then loading and verifing another data pattern from the stop to start addresses (reverse addressing).

4.3.13 Test 13--Moving Inversions With Seed X'55555555'

This test executes a read/write/read to every location between the start and stop addresses. Addressing is executed in both the forward and reverse directions. The starting "seed" data pattern is X'55555555'.

The test loads memory with zeros from the start and stop addresses by first loading and verifying a data pattern from the start to stop addresses (forward addressing), then loading and verifying another data pattern from the stop to start addresses (reverse addressing).

4.3.14 Test 14--Moving Inversions With Seed X'AAAAAAAA' 2 Word Addressing Increment

This test executes a read/write/read to every location between the start and stop addresses. Addressing is executed in both the forward and reverse directions. The starting "seed" data pattern is X'AAAAAAAA'.

The test loads memory with zeros from the start and stop addresses by first loading and verifying a data pattern from the start to stop addresses (forward addressing), then loading and verifying another data pattern from the stop to start addresses (reverse addressing).

4.3.15 Test 15--Moving Inversions With Seed X'AAAAAAAA' 4 Word Addressing Increment

This test executes a read/write/read to every location between the stop and start addresses. Addressing is executed in both the forward and reverse directions. The starting "seed" data pattern is X'AAAAAAAA'.

The test loads memory with zeros from the start and stop addresses by first loading and verifying a data pattern from the start to stop addresses (forward addressing), then loading and verifying another data pattern from the stop to start addresses (reverse addressing).

4.3.16 Test 16—F and C Bit Test STB/LW

This test performs a store byte instruction at the current memory test location then loads a memory word to verify that the correct byte field is stored.

Note: All memory fetches on CONCEPT 32/87 are doubleword. All memory fetches on CONCEPT 32/67 are doubleword except when the shared memory controller (SMC) instructon is enabled. When the SMC instruction is enabled, only one word is fetched.

4.3.17 Test 17—F and C Bit Test STH/LW

This test performs a store halfword instruction at the current memory test location then loads a memory word to verify that the correct byte field is stored.

Note: All memory fetches on CONCEPT 32/87 are doubleword. All memory fetches on CONCEPT 32/67 are doubleword except when the shared memory controller (SMC) instructon is enabled. When the SMC instruction is enabled, only one word is fetched.

4.3.18 Test 18—F and C Bit Test STW/LB

This test performs a store word instruction at the current memory test location then loads a memory byte to verify that the correct byte field is stored.

Note: All memory fetches on CONCEPT 32/87 are doubleword. All memory fetches on CONCEPT 32/67 are doubleword except when the shared memory controller (SMC) instructon is enabled. When the SMC instruction is enabled, only one word is fetched.

4.3.19 Test 19—F and C Bit Test STW/LH

This test performs a store word instruction at the current memory test location then loads a memory halfword to verify that the correct byte field is stored.

Note: All memory fetches on CONCEPT 32/87 are doubleword. All memory fetches on CONCEPT 32/67 are doubleword except when the shared memory controller (SMC) instructon is enabled. When the SMC instruction is enabled, only one word is fetched.

4.3.20 Test 20--F and C Bit Test STD/LW

This test performs a store doubleword instruction at the current memory test location then loads a memory word to verify that the correct byte field is stored.

Note: All memory fetches on CONCEPT 32/87 are doubleword. All memory fetches on CONCEPT 32/67 are doubleword except when the shared memory controller (SMC) instructon is enabled. When the SMC instruction is enabled, only one word is fetched.

4.3.21 Test 21--F and C Bit Test STW/LD

This test performs a store doubleword instruction at the current memory test location then loads a doubleword to verify that the correct byte field is stored.

4.3.22 Test 22--Error Correction Code Check Using ECC Bit 14

This test verifies the ECC logic by inverting and verifying data bit 14 and checking for parity errors. If a parity error or data error occurs, the test reports the error.

4.3.23 Test 23--Error Correction Code Check Using ECC Bit 18

This test verifies the ECC logic by inverting and verifying data bit 18 and checking for parity errors. If a parity error or data error occurs, the test reports the error.

4.3.24 Test 24--Error Correction Code Check Using ECC Bit 19

This test verifies the ECC logic by inverting and verifying data bit 19 and checking for parity errors. If a parity error or data error occurs, the test reports the error.

4.4 Multiprocessor Type Tests

If one or more secondary processor is configured following execution of data pattern type tests, all processors enter the Multiprocessor Type Testing mode. The primary processor, then all secondary processors, execute tests 1 through 5 sequentially. All processors execute tests 6 through 13 concurrently.

4.4.1 Test 1--Memory Execution Test (Instruction Fetch - Path)

This test verifies the ability of the secondary processors to execute instructions in shared-memory. The program executes this test in a consecutive 192-instruction add bit in register (ABR) mini-program.

The primary processor copies this mini-program to shared memory in 2K-word map block boundaries from start address to end address of common shared-memory (randomly selected in 2K-words block after first pass). The test causes secondary processors to execute a branch and link (BL) instruction to each segment by primary processor which times out if a secondary processor takes too long to execute.

4.4.2 Test 2--Cache Write-thru Capability - Primary On

This test verifies the cache write-thru capability to shared memory with multiprocessor configuration. The secondary processor initializes a segment of shared memory with fixed patterns and waits. The primary processor then writes to that same segment of shared memory and thus invalidates the secondary processor's cache memory. The secondary processor then is scheduled to check for invalidation of it's cache by reading from shared memory.

This test executes with the primary processor's cache memory on.

4.4.3 Test 3--Cache Write-thru Capability - Primary Off

This test verifies the cache write-thru capability to shared memory with multiprocessor configuration. The secondary processor initializes a segment of shared memory with fixed data patterns and waits. The primary processor then writes to that same segment of shared memory and thus invalidates the secondary precessor's cache memory. The secondary processor then is scheduled to check for invalidation of its cache by reading from shared memory.

This test executes with primary processor's cache memory off.

4.4.4 Test 4--Cache Write-thru Capability - Secondary On

This test verifies the cache write-thru capability to shared memory with multiprocessor configuration. The primary processor initializes a segment of shared memory with fixed data patterns and waits. The secondary processor then writes to that same segment of shared memory and thus invalidates the primary processor's cache memory. The primary processor then is scheduled to check for invalidation of its cache by reading from shared memory.

This test executes wtih the secondary processor's cache memory on.

4.4.5 Test 5--Cache Write-thru Capability - Secondary Off

This test verifies the cache write-thru capability to shared memory with multiprocessor configuration. The primary processor initializes a segment of shared memory with fixed data patterns and waits. The secondary processor then writes to that same segment of shared memory and thus invalidates the primary processor's cache. The primary processor then is scheduled to check for invalidation of its cache by reading from shared memory.

This test executes with the secondary processor's cache memory off.

4.4.6 Test 6--Memory Execution Test With Contention (Instruction Fetch - Path) - Secondary

This test verifies secondary processors can execute instructions in shared-memory, which is dependent upon correct instruction fetches, while competing with each other. This test is performed in a consecutive 192-instruction add bit in register (ABR) mini-program copied by the primary processor to shared memory in 2K-word map block boundaries from start address to end address of common shared-memory (randomly selected in 2K block after first pass). Secondary processors then are scheduled to execute a branch and link (BL) instruction to each segment by the primary processor which will "time out" if a secondary takes too long to execute.

4.4.7 Test 7-- Memory Execution Test With Contention (Instruction Fetch - Path) - Primary

This test verifies primary and secondary processors can execute instructions in shared memory, which is dependent upon correct instruction fetches, while competing with each other. The program executes this test in a consecutive 192-instruction add bit in register (ABR) mini-program copied by the primary processor to shared memory in 2K-word map block boundaries from start address to end address of common shared memory (randomly selected in 2K blocks after first pass).

The test schedules secondary processors to execute a branch and link (BL) instruction to each segment by primary processor which times out if a secondary processor takes too long to execute. The primary processor then executes a branch and link (BL) instruction to the same segment of shared memory and verifies all processors were able to execute instructions in shared memory.

4.4.8 Test 8--Memory Contention Test (Read/Lock Interlock Test) - Secondary

This test creates contention between ports, controllers, and storage boards. The test executes from the start address to the end address of common shared memory. Each secondary processor executes a set bit in memory (SBM) loop to all locations. Each secondary processor, in its loop, sets a bit that corresponds to the sequence in which it was initialized.

The primary processor times out any secondary processor that cannot resolve lock/unlock contention. After all secondary processors execute their loop or time out, the primary processor reads from the common shared memory and checks whether each secondary processor accessed each word. Each secondary processor also reads from common shared memory and verifies that its individual bit is set.

4.4.9 Test 9--Memory Contention Test (Read/Lock Interlock Test) - Primary

This test creates contention between ports, controllers, and storage boards. The test executes from the start address to the end address of common shared memory. Each secondary processor then executes a set bit in memory (SBM) loop to all locations. After the secondary processors begin execution, the primary processor starts to execute an identical SBM loop addressing the same shared memory. Each processor sets a bit that corresponds to the sequence in which it was initialized.

The primary processor times out any secondary processor that cannot resolve lock/unlock contention. After all secondary processors execute their loop or time out, the primary processor reads from common shared memory and determines whether the primary and each secondary processor have accessed each word. Each secondary processor also reads from common shared memory and verifies its individual bit is set.

4.4.10 Test 10--Memory Contention Test (Zero/Set Bit in Memory) With Iteration

This test creates contention between ports, controllers, and storage boards. This test executes from the start address to the end address of common shared-memory. Each secondary processor then executes a set bit in memory (SMB) loop to all locations. After the secondary processors begin execution, the primary processor starts to execute an identical SMB loop addressing the same shared memory. Each secondary processor, in its loop, sets a bit that corresponds to the sequence in which it was initialized. All processors then execute a zero bit in memory (ZBM) loop to all locations. Each loop executes three times on each processor.

The primary processor times out any secondary processor that cannot resolve lock/unlock contention. After all secondary processors execute their loop or time out, the primary processor reads from the common shared memory and checks whether each secondary processor accessed each word. Each secondary processor also reads from common shared memory and verifies its individual bit is set.

4.4.11 Test 11--Memory Contention Test (Set Bit in Memory) - Primary Cache On

This test creates contention between ports, controllers, and storage boards. This test executes from the start address to the end address of common shared-memory. Each secondary processor executes a set bit memory (SMB) loop. After secondary processors begin execution, primary starts to execute an identical SBM loop addressing the same shared memory. Each processor sets a bit that corresponds to the sequence in which it was initialized.

The primary processor times out any secondary processor that cannot resolve lock/unlock contention. After all secondary processors execute their loop or time out, the primary processor reads from the common shared memory and checks whether each secondary processor as well as the primary processor, accessed each word. Then each secondary processor reads from common shared memory and verifies its individual bit is set.

Note: This test is executed with the primary processor's and the secondary processor's cache on.

4.4.12 Test 12--Critical Section Test - Cache Off

This test creates contention between ports, controllers, and storage boards. The primary processor schedules the secondary processors to concurrently simulate a typical semaphore setting to gain entry to a critical section of memory. The primary processor initializes the semaphore with a set bit in memory (SBM) instruction to a word in shared memory. Each secondary processor then executes a SBM-branch-to-self loop, attempting to gain entry to the critical section. Once the critical section has been entered, each secondary processor executes a zero bit in memory (ZMB) instruction to unlock the semaphore. This is repeated to every word of an 8-word block on every 2K-word map boundary (randomly selected after the first pass).

4.4.13 Test 13--Critical Section Test - Cache On

This test creates contention between ports, controllers, and storage boards. The primary processor schedules the secondary processors to concurrently simulate a typical semaphore setting to gain entry to a critical section of memory. The primary processor initializes the semaphore with a set bit in memory (SBM) instruction to a word in shared memory. Each secondary processor then executes a SBM-branch-to-self loop, attempting to gain entry to the critical section. Once the critical section has been entered, each secondary processor executes a zero bit in memory (ZMB) instruction to unlock the semaphore. This is repeated to every word of an 8-word block on every 2K-word map boundary (randomly selected after the first pass).

Note: This test is executed with primary processor's and the secondary processor's cache on.

4.5 Manual Intervention Tests

The primary processor executes the following tests once if shared memory is present. These tests execute during the first pass of the program and require operator intervention. Inhibit execution of these tests by setting option 8 (inhibit operator intervention tests).

The program writes the following message to the primary processor's operator's console prior to execution of manual intervention tests:

```
BEGIN EXECUTION OF MANUAL INTERVENTION TESTS
```

4.5.1 Manual Intervention Test 1—Battery Back-up

This test verifies shared memory is able to retain data after a power down on shared memory ((MS)2) with battery back-up.

The test writes an address in address data pattern to the specified shared memory boundary ranges. The program writes out the following messages:

```
TURN SHARED MEMORY POWER SWITCH OFF...
THEN TURN SHARED MEMORY POWER SWITCH ON AND RESYNCHRONIZE
CLOCKS. ON (MS)2 SYSTEM PRESS RESET ON CONTROL PANEL.
PRESS CR WHEN READY.
```

4.5.2 Manual Intervention Test 2--ECC Switches

This test verifies the ECC switches are functioning properly. Prior to execution of this test, the test issues the following prompt to the primary processor's operator's console:

```
ENTER "MS 2" IF SHARED MEMORY IS (MS)2 MODEL
DEFAULT OF CR IF NOT.
```

When default is specified, this test exits. If shared memory is a Multiprocessor Shared Memory System, the test writes the following messages:

```
ON (MS)2 SYSTEMS THIS TEST CHECKS
IF PARITY ERROR OCCURS.
IF ALL ERROR SWITCH IS ENGAGED,
SWITCH ON ONLY ONE OF THE ECC SWITCHES
OTHERWISE SWITCH ON ANY COMBINATION OF TWO SWITCHES.
PRESS CR WHEN READY.
```

The test executes a write/read data pattern to every location of shared memory and a parity error is expected in each location.

If parity errors are reported in every location, the test writes the following message to the primary processor's operator's console:

```
CONTINUE TEST ECC SWITCH?
ENTER "N" FOR NO OR JUST DEFAULT
OF CR FOR YES.
```

Responding with an N (no) causes the test to write the following message to the primary processor's operator's console:

```
WHEN EXITING THIS TEST MAKE CERTAIN ALL ECC SWITCHES ARE
ENGAGED AND "CLRST" SWITCH HAS BEEN ENGAGED ONCE.
PRESS CR WHEN READY.>
```

If no further ECC switch testing is necessary, return ECC switches to their original position.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of the Gould CONCEPT/32 Multiprocessor Shared Memory Diagnostic (CN.MSM.r):

```
LOAD CN.MSM.r
*LOADED: 'CN.MSM.r'
PGM: 03000
START CN.MSM.r PASS(1)
START CN.MSM.r (#111031-253r date)
CONCEPT/32 MULTIPROCESSOR SHARED MEMORY DIAG
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
OPTIONS REGISTER (LOC 00003044) = 22811000
TEST EXECUTION CONTROL LIST (LOC 00003100) = 0000000007.

NOTE: INSURE FOLLOWING SWITCHES ARE ENGAGED (SWITCHED TO LEFT) ON
      SHARED MEMORY SYSTEM:
MEMORY CONFIGURATION:
(000000-7FFFFC)     (2048.KW)       PRESENT
(800000-FFFFFC)     (2048.KW)   NONPRESENT

IS THIS DIAGNOSTIC IMAGE LOADED IN A PRIMARY CPU
OR A SECONDARY CPU (NODE) ?
ENTER "P" FOR PRIMARY OR JUST CR FOR DEFAULT OF SECONDARY NODE > P

THIS DIAGNOSTIC IS LOADED ON A PRIMARY PROCESSOR.

PRIMARY PROCESSOR'S PRIVATE MEMORY START ADDRESS ?
ENTER "0" FOR NO PRIVATE MEMORY TESTING THEN CR OR
ENTER HEX.  WORD ADDRESS THEN CR OR
JUST CR FOR DEFAULT OF 00019BA0 >

PRIMARY PROCESSOR'S PRIVATE MEMORY END ADDRESS ?
ENTER HEX. WORD ADDRESS THEN CR OR
JUST CR FOR DEFAULT OF 00049FFC >

PRIMARY PROCESSOR'S SHARED MEMORY START ADDRESS ?
ENTER "0" FOR NO SHARED MEMORY TESTING THEN CR OR
ENTER HEX. WORD ADDRESS THEN CR OR
JUST CR FOR DEFAULT OF 00050000 >

PRIMARY PROCESSOR'S SHARED MEMORY END ADDRESS ?
ENTER HEX. WORD ADDRESS THEN CR OR
JUST CR FOR DEFAULT OF 007FFFFC >
```

```
HOW MANY SECONDARY CPUS (NODES) ARE CONFIGURED ?
ENTER DECIMAL NUMBER INCLUSIVELY BETWEEN 0 AND 15 THEN CR
OR JUST CR FOR DEFAULT OF 0 >

BEGIN EXECUTION OF DATA-PATTERN-TYPE TESTS.
TEST01. STORE DATA PATTERN USING ALL ZEROS.
TEST02. STORE DATA PATTERN USING ALL ONES.
TEST03. STORE DATA PATTERN USING ALL ALTERNATING ONES : X'55555555'.
TEST04. STORE DATA PATTERN USING ALL ALTERNATING ONES : X'AAAAAAAA'.
TEST05. STORE DATA PATTERN USING ADDRESS IN ADDRESS.
TEST06. STORE DATA PATTERN FOR REFRESH USING ALL ONES.
TEST07. STORE DATA PATTERN FOR REFRESH USING ALL ZEROS.
TEST08. STORE DATA PATTERN FOR REFRESH USING ALTERNATING ONES : X'55555555'.
TEST09. STORE DATA PATTERN FOR REFRESH USING ALTERNATING ONES : X'AAAAAAAA'.
TEST10. STORE DATA PATTERN FOR REFRESH USING ADDRESS IN ADDRESS.
TEST11. MOVING INVERSIONS TEST SEED = X'11111111'.
TEST12. MOVING INVERSIONS TEST SEED = X'AAAAAAAA'.
TEST13. MOVING INVERSIONS TEST SEED = X'55555555'.
TEST14. MOVING INVERSIONS TEST SEED = X'AAAAAAAA'- 2 WORD INCREM.
TEST15. MOVING INVERSIONS TEST SEED = X'AAAAAAAA'- 4 WORD INCREM.
TEST16. F & C BITS TEST STB/LW .
TEST17. F & C BITS TEST STH/LW .
TEST18. F & C BITS TEST STW/LB .
TEST19. F & C BITS TEST STW/LH .
TEST20. F & C BITS TEST STD/LW .
TEST21. F & C BITS TEST STW/LD .
TEST22. ERROR CORRECTION CODE CHECK DATA BIT 14.
TEST23. ERROR CORRECTION CODE CHECK DATA BIT 18.
TEST24. ERROR CORRECTION CODE CHECK DATA BIT 19.

END PASS: 1
  (ERRORS: 0.SINCE LAST REPORT, 0.SINCE PROG START)

END CN.MSM.r: 1 PASS(ES)
```

Gould CONCEPT/32™

Trap

(CN.TRP.r)

Diagnostic Description

Diagnostic Number: 111031-241r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005550-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Trap (CN.TRP.r) Diagnostic Description Release 1.0, Publication Order Number **326-005550-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-6
3-1 through 3-3/3-4
4-1 through 4-6
5-1 through 5-5/5-6

CONTENTS

CHAPTER 1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1
1.3 Execution Time 1-1

CHAPTER 2 PROGRAM EXECUTION

2.1 General Description 2-1
2.2 Option Set 2-2
2.3 CPU Execution Modes 2-3
2.4 Execution Map Configuration 2-4
2.5 Memory Configuration 2-4
2.6 Trap and Interrupt Configuration 2-4

CHAPTER 3 ERROR REPORTING

3.1 General Description 3-1
3.2 Error Message Format 3-2
3.3 Optional Programmed Halt 3-3

CHAPTER 4 TEST DESCRIPTIONS

4.1 Summary of Tests 4-1
4.2 Test 1--Halt Trap (PL0E) 4-1
4.3 Test 2--Attention Interrupt (PL0D) 4-1
4.4 Test 3--Power Fail Safe/Auto Start Trap (PL00,PL01) 4-2
4.5 Test 4--Address Specification Trap (PL0C) 4-2
4.6 Test 5--Nonpresent Memory Trap (PL03) 4-3
4.7 Test 6--Undefined Instruction Trap (PL04) 4-3
4.8 Test 7--Privilege Violation Trap (PL05) 4-4
4.9 Test 8--Real-Time Clock Interrupt (PL18) 4-4
4.10 Test 9--Arithmetic Exception Trap (PL0F) 4-4
4.11 Test 10--System Check Trap (PL08) 4-5
4.12 Test 11--Supervisor Call Trap (PL06) 4-5
4.13 Test 12--Block Mode Activation and Control Trap 4-6
4.14 Test 13--Machine Check Trap (PL07) 4-6
4.15 Test 14--Call Monitor Trap (PL0A) 4-6

CHAPTER 5 EXAMPLE EXECUTION 5-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

Gould CONCEPT/32 Trap Diagnostic (CN.TRP.r) is a level one macrodiagnostic program that verifies the context switching mechanism of all software accessible traps. In addition, this program verifies the vector location, stored status doubleword, and the relative priority level on each trap occurrence.

Options have been implemented that provide direct operator control over stop on error, loop on error, loop on test and loop on subtest.

Note: This manual describes the specific attributes of Gould CONCEPT/32 Trap Diagnostic (CN.TRP.r). For a discussion of level one macrodiagnostic attributes, including the level one macrodiagnostic load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

1.2 Resource Requirements

This program requires a CONCEPT/32 CPU with a minimum of 64K words of memory and either of the following:

- One to six Model 2345 Real-Time Option Modules
- One IOP Model 8000 or 8001

The program resides at a default starting address of 3000_{16}. This program must not be loaded into any memory module other than zero.

In addition to program storage in main memory, this program requires a 2000_{16} word interrupt dispatch queue.

The channel address and device subaddress of the operator's console must be $7EFC_{16}$.

1.3 Execution Time

This program requires approximately 5 to 20 seconds to execute the first pass and 10 to 30 seconds to execute each subsequent 10 passes depending upon the following:

- machine type
- machine configuration
- option setting

CHAPTER 2

PROGRAM EXECUTION

2.1 General Description

Gould CONCEPT/32 Trap Diagnostic (CN.TRP.r) follows the load and start procedures established for all level one macrodiagnostic programs. The following are the four steps to level one macrodiagnostic program execution:

1. IPL the macrodiagnostic facility
2. Load the level one macrodiagnostic program
3. Set or reset options
4. Start program execution

For a complete description of the level one macrodiagnostic program load and start procedures, refer to the CONCEPT/32 Diagnostic Facility User's Manual.

Upon completion of the initial program load (IPL) sequence and the load and start procedures, the program writes the following start message to the operator's console:

```
START CN.TRP.r (#111031-241r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

For an example execution sequence, refer to chapter 5.

2.2 Option Set

The following options have been implemented for this program:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If set during an error message report, the program completes the current message before setting the option. This option does not inhibit the function defined by option 1.
1	Reset	Rings the bell on the operator's console once per error.
2	Set	Executes a programmed halt upon error detection. Unless inhibited by option 0, the program writes the message associated with the error prior to the halt. Nullify the programmed halt by entering the Panel mode Run command on the operator's console.
3	Reset	Repeats an error producing test or subtest until the test or subtest executes error free. Operand values cannot be modified during the repeat function. If set, the program executes the functions defined by options 0, 1, and 2 before repeating a test or subtest.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire program until the option is reset or a predetermined pass count is reached. Pass count can be specified using the Diagnostic Overlay Loader Start command.
7	Reset	Unimplemented.
8	Reset	Inhibits execution of all tests requiring operator intervention.
9	Reset	Unimplemented.
10	Set	Writes the pass count messages to the operator's console upon completion of each pass.
11	Set	Writes the milestone messages to the line printer.
12	Reset	Inhibits Extended Addressing mode. Perform extended addressing on every pass following pass one by resetting this option.
13	Reset	Inhibits Mapped mode. Set Mapped mode for every pass following pass two by resetting this option.
14	Reset	Inhibits interval timer. Activate the interval timer for every pass following pass three by setting this option.

Option	Default	Effect When Set
15-16	Reset	Unimplemented.
17	Reset	Writes map configuration dump.
18	Reset	Unimplemented.
19	Reset	Echoes all messages to the line printer.
20	Reset	Unimplemented.
21	Reset	Writes scratchpad image.
22-25	Reset	Unimplemented.
26	Reset	Causes the program to immediately halt upon error detection to allow for a snapshot of the error location. The program does not write an error message or save the contents of the GPRs. Nullify the halt by entering the Panel mode Run command on the operators console. When this option and option 2 are both set, the function defined by this option executes before the function defined by option 2.
27-31	Reset	Reserved - Do not use.

2.3 CPU Execution Modes

The CONCEPT/32 Trap Diagnostic (CN.TRP.r) tests all CONCEPT/32 CPU execution modes. The following are the execution modes under default conditions:

Pass #	Extended Mode	Mapped Mode	Interval Timer
1	No	No	No
2	Yes	No	No
3	Yes	Yes	No
4-n	Yes	Yes	Yes

2.4 Execution Map Configuration

The program sets up the execution map configuration with memory module zero (map registers 00-04) marked valid. The program marks all other map entries invalid and nonpresent ($7FF_{16}$). All tests run in the Mapped mode use this execution map configuration.

The program configures the execution map with all map registers in a one-to-one mapping relationship with all map blocks.

2.5 Memory Configuration

The program dynamically determines memory size (starting at location zero) and writes the following message to the operator's console:

```
MEMORY CONFIGURATION:

(aaaaaa-aaaaaa) (mmmmKW)          PRESENT
(aaaaaa-aaaaaa) (mmmmKW)          NONPRESENT
```

aaaaaa — specifies addresses making up a range in memory that is either present or nonpresent.

mmmm — specifies the amount of memory in kilowords (KW) that makes up an address range.

2.6 Trap and Interrupt Configuration

The program determines the trap and interrupt configuration present by directing Request Interrupt (RI) instructions at priority levels 00-6F. A priority level is defined as being present if an interrupt is received.

While scanning all priority levels, the program checks for the presence of the following interrupts:

- An interrupt received from an enabled level before requested
- An interrupt received from a priority level whose dispatch linkage queue vector indicates usage of a nonstandard dedicated location
- An interrupt level differing from the requested interrupt level

If any of these interrupts exist, the program disables the priority level and writes a message indicating that the priority level is skipped and the reason. The program removes the level from the list of available priority levels.

If the level received differs from the level requested, it could indicate an error in the scratchpad configuration, or that more than one interrupt is connected to a device.

The program assigns priorities to all levels once it determines that all levels are present. The program then writes out a log of all priority levels present, in order of priority, with an indication of priority levels skipped.

The following is a typical configuration of traps and interrupts from a system with one

The following is a typical configuration of traps and interrupts from a system with one RTOM or an IOP Controller. If option 11 (write milestone message) is reset, the following message is inhibited.

Levels Present (By Priority)

PL	DED-L	LNK-L	TYPE	CLASS	ADDR
00	00080	0645C	TRAP		
01	00084	06480	TRAP		
02	00088	064A4	TRAP		
03	0008C	064C8	TRAP		
04	00090	064EC	TRAP		
05	00094	06510	TRAP		
06	00098	06534	TRAP		
07	0009C	06558	TRAP		
08	000A0	0657C	TRAP		
09	000A4	065A0	TRAP		
0A	000A8	075C4	TRAP		
0B	000AC	065E8	TRAP		
0C	000B0	0660C	TRAP		
0D	000B4	06630	TRAP		
0E	000B8	06654	TRAP		
0F	000BC	06678	TRAP		
00	00100	068DC	EXT		
01	00104	06900	EXT		
02	00108	06924	EXT		
03	0010C	06948	EXT		
10	00140	06B1C	I/O		
11	00144	06B40	I/O		
13	0014C	06B88	I/O	F	7E
14	00150	06BAC	EXT		
15	00154	06BD0	EXT		
16	00158	06BF4	EXT		
17	0015C	06C18	EXT		
18	00160	06C3C	EXT		
19	00164	06C60	EXT		
1A	00168	06C84	EXT		
1B	0016C	06CA8	I/O	3	7F
1C	00170	06CCC	EXT		
1D	00174	06CF0	EXT		
1E	00178	06D14	EXT		
1F	0017C	06D38	EXT		

DED-L — indicates the priority level's system reset dedicated location.

LNK-L — indicates the address of the nine word interrupt context block used to field the interrupt or trap.

TYPE — indicates either a trap, external interrupt, or I/O interrupt.

CLASS and ADDR — indicate the data contained in the scratchpad entry for I/O devices including Class F devices.

If a dynamic request occurs, the program appends the following message to the priority level message:

,SKIPPED (APPARENTLY DYNAMIC)

If a requested interrupt level vectors through a nonstandard dedicated location, the program appends the following message to the priority level message:

,SKIPPED (NON-STD DED-L)

CHAPTER 3

ERROR REPORTING

3.1 General Description

If an error condition occurs during the execution of Gould CONCEPT/32 Trap Diagnostic (CN.TRP.r), and the default option settings are in effect, the program writes an error message to the operator's console and halts execution. The Halt instruction, called by the program, places the operator's console in Panel mode.

Refer to Section 3.2 for a description of the standard format for error messages generated during execution of this program.

To recover from an error condition, issue the Run command on the operator's console. This command causes the program to continue execution at a location immediately following the location where the error occurred. If in subsequent passes the error continues to occur, refer to the source code listing to determine the cause of the error.

If the error condition fails to occur on subsequent passes of the program, restart the program. If the error condition fails to occur after a restart, no error condition exists.

3.2 Error Message Format

Errors genrated during the execution of this program write out one or more of the following messages:

{NO / ONE / xxxx} {INTRPT(S) / TRAP(S)} REC FROM PLyy AT aaaaa, {PLzz EXP / EXP AT bbbbb / PLzz EXP AT bbbbb / NO {INTRPT(S) / TRAP(S)} EXP}

NO	indicates that no interrupt or trap was received.
ONE	indicates that one and only one interrupt or trap was received.
xxxx	specifies the number of interrupts or traps received if other than NO or ONE interrupt or trap was received.
PLyy	indicates either the priority level of the received interrupt or trap if ONE or xxxx interrupts were received or indicates the priority level of the expected interrupt or trap if NO interrupt or trap was received.
aaaaa	specifies the actual address at the point of interrupt/trap if one or more interrupts/traps were received or specifies the expected address at the point of interrupt/trap if no interrupt/trap was received but one was expected.
PLzz	indicates the priority level of the interrupt or trap expected.
bbbbb	specifies the expected address at the point of interrupt/trap if an interrupt/trap was received and aaaaa does not equal bbbbb.
NO {INTRPT(S) / TRAP(S)} EXP	indicates that no interrupt or trap was expected.

All addresses are expressed as program listing vectors.

3.3 Optional Programmed Halt

If option 2 (halt on error) is set, the general purpose registers are formatted prior to the programmed halt as follows:

- GPR0 Memory relative address of the call to the CHECKXX routine that decoded the error.
- GPR1 Number of interrupts received on a single request.
- GPR2 Defined by the test sequence.
- GPR4 Listing vector address of the call to the CHECKXX routine that decoded the error.
- GPR6 Contains the values for the priority level (bits 0-7) and the interrupt/trap address (bits 13-31) for an interrupt/trap.
- GPR7 Contains the actual hardware values for the priority level (bits 0-7) and the interrupt/trap address (bits 13-31) for an interrupt/trap received.

Option 2 does not affect GPR 3 and 5.

Rules for Optional Programmed Halt

- If the error associated with the failure indicates that no interrupt has been received and GPR1=0, the contents of GPR7 contains information pertaining to the last interrupt serviced.
- Any error message not covered above can be found in the description of the test to which it pertains.

CHAPTER 4

TEST DESCRIPTIONS

4.1 Summary of Tests

Gould CONCEPT/32 Trap Diagnostic (CN.TRP.r) contains 14 tests which check the context switching mechanism of all software accessible traps. Each of the following tests verify the unique properties of individual traps.

4.2 Test 1—Halt Trap (PL0E)

This test verifies that a Halt instruction causes a Halt Trap when it is executed in the privileged mode with the enable Halt Trap bit (23) set in the CPU status word.

4.3 Test 2—Attention Interrupt (PL0D)

This test verifies the receipt of an attention trap from the operator's console. The test verifies a dynamic external request by issuing the following prompt:

ISSUE 'ATTENTION' (BELL)

Respond to the prompt by entering @@A (ATTENTION). If no response is given within 10 seconds, the program writes an error message.

4.4 Test 3—Power Fail Safe/Auto Start Trap (PL00,PL01)

This test verifies the Power Fail Safe/Auto Start trap operation. The test writes the following message to the operator's console:

```
..POWER DOWN, WAIT, AND RESTORE POWER (BELL)
..CAUTION - BATTERY EQUIPPED MOS SYSTEMS ONLY.
        - MUST DISCONNECT A/C ON BATTERY EQUIPPED SYSTEMS
.. IF POWER DOWN IS NOT FEASIBLE, ISSUE "ATTENTION" TO ESCAPE
IGNORE ERRORS AND CONTINUE THE PROGRAM (RUN).
```

When power is restored, recovery of the program and the following message constitute success:

```
POWER DOWN EXECUTION TIME .xxxMS
```

The time (.xxxMS) indicates the processing time available during a power failure.

This test also verifies that a power fail does not cause a parity error to be written to memory. In the event that a parity error is written to memory on a power fail, the program writes the following message to the operator's console:

```
..PARITY ERROR GENERATED ON POWER DOWN
```

In addition, the test verifies that the CPU scratchpad is correctly stored and retrieved during the power fail sequence. The test checks a rolled-out scratchpad against a copy of the pre-power down scratchpad, then checks a rolled-in scratchpad for correct data and for the reset state of interrupt and device flags. This check ensures that the correct firmware sequences have been entered for scratchpad roll-in and initialization during power up and that no spurious power down interrupts have been received while powering up.

If a scratchpad error occurs during power fail, the test writes the following message to the operator's console:

```
ROLL-IN/OUT ERROR DURING POWER
UP/DOWN AT *****
SCRATCHPAD ADDRESS: **
EXP: ********
ACT: ********
```

4.5 Test 4—Address Specification Trap (PL0C)

This test verifies that an Address Specification trap occurs when an illegal address or register is used within an instruction.

4.6 Test 5—Nonpresent Memory Trap (PL03)

This test verifies the Nonpresent Memory trap.

The test locates the first nonpresent memory module in the lower 128K and uses it to check the following instructions:

BU	Branch to Nonpresent Memory
TRSW	Transfer to Nonpresent Memory
EXM	Execute Nonpresent Memory
LW	Operand Fetch From Nonpresent Memory
STW	Operand Store Into Nonpresent Memory

Each of the instructions (except TRSW) executes using direct, indirect, pre-indexed, and indirect with post-index access techniques. If an error occurs, the receiver address is the last address in the nonpresent memory tested.

The test verifies proper execution by causing a Branch and Link (BL) to be executed from the highest present memory address.

The test verifies nonpresent memory by causing the Load Map (LMAP), and Load Program Status Doubleword and Change Memory (LPSDCM) instructions to be executed with their map image pointers pointing to nonpresent memory.

4.7 Test 6—Undefined Instruction Trap (PL04)

This test verifies the Undefined Instruction trap by executing one of the following operations:

- Executing each of the halfword operation codes in the left halfword position followed by a No Operation (NOP) instruction.
- Executing each of the halfword operation codes in the right halfword position preceded by a NOP instruction.
- Executing each of the defined fullword operation codes in the right halfword position preceded by a NOP instruction.

Any of the four following conditions trigger the Undefined Instruction trap:

- Undefined instructions
- A Call Monitor (CALM) in the right halfword or CALM instructions back-to-back
- Performing an Execute Memory (EXM), Execute Register (EXR) or Execute Register Right (EXRR) on an execute (EXM, EXR, EXRR) instruction
- Class F I/O instructions with illegal subop codes

4.8 Test 7—Privilege Violation Trap (PL05)

This test verifies the Privilege Violation trap by executing the privileged and unprivileged instruction set in the Unprivileged mode.

4.9 Test 8—Real-Time Clock Interrupt (PL18)

This test verifies the Real-Time Clock interrupt rate. The test also verifies that no Real-Time Clock interrupts are received while Clock Override is in effect.

The program issues the following prompt to the operator's console:

...TURN CLOCK OVERRIDE ON

Enter OVR(cr) to turn clock override on.

The program waits for 5 seconds to accumulate the number of Real-Time Clock interrupts received. The following message indicates a Clock Override failure:

INTRPT(S) REC FROM PL18, NONE EXPECTED

Upon completion of the test, the program issues the following prompt to the operator's console:

...TURN CLOCK OVERRIDE OFF

Enter OVR(cr) to turn Clock Override off.

When set, option 8 (inhibit operator intervention tests) causes the program to skip this test.

4.10 Test 9—Arithmetic Exception Trap (PL0F)

This test verifies the Arithmetic Exception trap by executing instructions which produce an overflow/underflow condition. This test checks the following classes of instructions:

- Fixed-Point Memory Reference Arithmetic
- Immediate Arithmetic
- Register-to-Register Arithmetic
- Add Register to Memory
- Add Bit in Memory (ABM)
- Add Bit in Register (ABR)
- Negate (LNW, LND, TRN, TRNM)
- Arithmetic Shift (SLA, SLAD)
- Arithmetic Round (RND)
- Execute Register (EXRR, EXR)
- Execute Memory (EXM)

4.11 Test 10—System Check Trap (PL08)

This test verifies the System Check trap using the following five software controllable cases:

- Test Device (TD) instruction to a Class F device
- Command Device (CD) instruction to a Class F device
- Write Channel Writable Control Store (WCWCS) instruction not preceded by an Enable Channel Writable Control Store (ECWCS) instruction
- WAIT instruction while in the Blocked mode
- Class F I/O instructions to non-class F devices (to device 2000_{16})

The test creates and stores a scratchpad entry at location $7B_{16}$ and executes all of the above instructions to an imaginary Class F I/O device at that location.

Example

```
TD X'7B', X'8000'.
```

This example assigns the Test Device (TD) instruction at the TD8000 test level (X'8000 test code) to scratchpad location $7B_{16}$ (X'7B).

4.12 Test 11—Supervisor Call Trap (PL06)

This test verifies the Supervisor Call Trap (SVC) by executing a base test through secondary vector zero then performing a comprehensive checkout with an SVC instruction indexing through all 16 possible secondary vector locations.

This test checks correct vectoring and correct call number storage by fetching a series of 16 halfword entries from a table in the following form:

```
0000
1111
 .
 .
FFFF
```

The test inserts these entries into bits 16-31 of the SVC instruction, which provides the secondary vector to index through (bits 16-19) and the call number to store (bits 20-31).

The program writes the following message to the operator's console if a secondary vector error occurs:

```
SECONDARY VECTOR ERROR AT *****
EXPECTED VECTOR: *
ACTUAL VECTOR: *
```

The program writes the following message to the operator's console if a call number error occurs:

```
CALL NUMBER ERROR AT *****
EXPECTED CALL #: ***
ACTUAL CALL #: ***
```

4.13 Test 12--Block Mode Activation and Control Trap

This test verifies the proper functioning of the Block External Interrupt, the Unblock External Interrupt, and the External Interrupt Flag bits of the Program Status Doubleword.

The test verifies the proper function of the Blocked External Interrupt and Unblocked External Interupt on all traps and interrupts by triggering each trap with interrupts blocked then triggering each interrupt.

The test verifies the proper functiong of the External Interrupt Flag bits of the PSD by executing a Load PSD macro with each configuration of the External Interrupt Flag bits. The test executes the Load PSD macro first with interrupts blocked then unblocked.

If an error in this test occurs, the program writes either of the following messages to the operator's console, depending upon the External Interrupt Flag bit configuration:

```
CPU FAILED TO LOAD NEW BLOCKING STATE AT *****
EIF: **
                                BLOCKED
CURRENT BLOCKING STATE:         UNBLOCKED
```

or

```
CPU FAILED TO RETAIN CURRENT BLOCKING STATE AT *****
EIF: **
                                BLOCKED
CURRENT BLOCKING STATE:         UNBLOCKED
```

4.14 Test 13--Machine Check Trap (PL07)

This test verifies the Machine Check trap using the following four software controllable cases:

- Attempting a Supervisor Call (SVC) trap with a nonpresent memory address in the master process list (MPL).
- Attempting an interrupt context switch with a nonpresent memory address in the master process list (MPL).
- Attempting a SVC trap with map enabled which forces a map overflow condition.
- Attempting an interrupt context switch with map enabled which forces a map overflow condition.

4.15 Test 14 -- Call Monitor Trap (PL0A)

This test verifies that the Call Monitor (CALM) instruction is functioning properly.

This test executes only for the 32/27 processor. The program bypasses this test for all other processor types.

CHAPTER 5

EXAMPLE EXECUTION

The following example displays a successful execution of Gould CONCEPT/32 Trap Diagnostic (CN.TRP.r).

```
>LOAD CN.TRP.r
*LOADED: 'CN.TRP.r'
 PGM: 03000
>STAR CN.TRP.r PASS(10)

START CN.TRP.r (#111031-241r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx

MEMORY CONFIGURATION:

(000000-1FFFFC)          ( 512KW)          PRESENT
(200000-FFFFFC)          (3584KW)      NON-PRESENT
LEVELS PRESENT (BY PRIORITY):
PL  DED-L  LNK-L  TYPE  CLASS  ADDR
00  00080  0BD94  TRAP
01  00084  0BDB8  TRAP
02  00088  0BDDC  TRAP
03  0008C  0BE00  TRAP
04  00090  0BE24  TRAP
05  00094  0BE48  TRAP
06  00098  0BE6C  TRAP
07  0009C  0BE90  TRAP
08  000A0  0BEB4  TRAP
09  000A4  0BED8  TRAP
0A  000A8  0BEFC  TRAP
0B  000AC  0BF20  TRAP
0C  000B0  0BF44  TRAP
0D  000B4  0BF68  TRAP
0E  000B8  0BF8C  TRAP
0F  000BC  0BFB0  TRAP
10  000C0  0BFD4  TRAP
00  00100  0C214  EXT
01  00104  0C238  EXT
02  00108  0C25C  EXT
03  0010C  0C280  EXT
```

```
06  00118  0C2EC I/O    F       08
08  00120  0C334 I/O    F       10
13  0014C  082F0 I/O    F       7E
14  00150  0C4E4 EXT
15  00154  0C508 EXT
16  00158  0C52C EXT
17  0015C  0C550 EXT
18  00160  0C574 EXT
19  00164  0C598 EXT
1A  00168  0C5BC EXT
1B  0016C  0C5E0 I/O    3       7F
1C  00170  0C604 EXT
1D  00174  0C628 EXT
1E  00178  0C64C EXT
1F  0017C  0C670 EXT
MODES: EXTENDED MAPPED INTERVAL-TIMER
            NO          NO          NO

TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
    MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
    MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
    MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)

END PASS:          1
MODES: EXTENDED MAPPED INTERVAL-TIMER
            YES       NO          NO
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
    MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
    MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
    MANUAL INTERVENTION TEST SKIPPED.
```

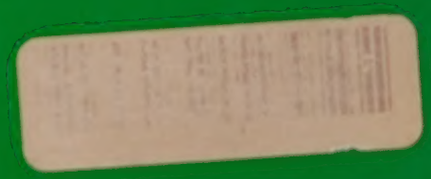
```
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)
MODES: EXTENDED MAPPED INTERVAL-TIMER
          YES      YES            NO
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
    MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
    MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
      MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)
MODES: EXTENDED MAPPED INTERVAL-TIMER
          YES      YES            YES
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
      MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
      MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
      MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
      MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
      MANUAL INTERVENTION TEST SKIPPED.
```

```
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 14 - CALL MONITOR TRAP (PL0A)
TEST  1 - HALT TRAP (PL0E)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
        MANUAL INTERVENTION TEST SKIPPED.
```

```
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)
TEST  1 - HALT TRAP (PL0E)
TEST  2 - CONSOLE (ATTENTION) TRAP (PL0D)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  3 - POWER FAIL/AUTO START (PL00/PL01)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  4 - ADDRESS SPECIFICATION (PL0C)
TEST  5 - NON-PRESENT MEMORY (PL03)
TEST  6 - UNDEFINED INSTRUCTIONS (PL04)
TEST  7 - PRIVILEGE VIOLATIONS (PL05)
TEST  8 - REAL TIME CLOCK (PL18)
        MANUAL INTERVENTION TEST SKIPPED.
TEST  9 - ARITHMETIC EXCEPTION (PL0F)
TEST 10 - SYSTEM CHECK TRAP (PL08)
TEST 11 - SUPERVISOR CALL TRAP (PL06)
TEST 12 - BLOCK MODE ACTIVATION AND CONTROL
TEST 13 - MACHINE CHECK TRAP (PL07)
TEST 14 - CALL MONITOR TRAP (PL0A)

END PASS:          10

END CN.TRP.r:                    10 PASS(ES)
```



Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd.
Fort Lauderdale, FL U.S.A. 33313-4499
Telephone (305) 587-2900

GOULD
Electronics